

2013/14

Universidad Pontificia Comillas

Escuela Superior de Ingeniería Comillas (ICAI)

Memoria descriptiva del proyecto fin de grado en
Ingeniería Electromecánica - electrónica



DESARROLLO DE UN PROTOTIPO DE ORDENADOR PARA BICICLETA DE MONTAÑA

DIRIGIDO POR: ANTONIO MUÑOZ SAN ROQUE & JOSÉ VILLAR COLLADO

IÑIGO FERNÁNDEZ DE ARAOZ GARCÍA - MIÑAUR



AUTORIZACIÓN PARA LA DIGITALIZACIÓN, DEPÓSITO Y DIVULGACIÓN EN ACCESO ABIERTO (RESTRINGIDO) DE DOCUMENTACIÓN

1º. Declaración de la autoría y acreditación de la misma.

El autor D. Íñigo María Fernández de Araoz García - Miñaur, como estudiante de Grado en Electromecánica de la UNIVERSIDAD PONTIFICIA COMILLAS (COMILLAS), DECLARA que es el titular de los derechos de propiedad intelectual, objeto de la presente cesión, en relación con el Proyecto de Fin de Grado titulado Desarrollo de un prototipo de ordenador para una bicicleta de montaña, que ésta es una obra original, y que ostenta la condición de autor en el sentido que otorga la Ley de Propiedad Intelectual como titular único o cotitular de la obra.

En caso de ser cotitular, el autor (firmante) declara asimismo que cuenta con el consentimiento de los restantes titulares para hacer la presente cesión. En caso de previa cesión a terceros de derechos de explotación de la obra, el autor declara que tiene la oportuna autorización de dichos titulares de derechos a los fines de esta cesión o bien que retiene la facultad de ceder estos derechos en la forma prevista en la presente cesión y así lo acredita.

2º. Objeto y fines de la cesión.

Con el fin de dar la máxima difusión a la obra citada a través del Repositorio institucional de la Universidad y hacer posible su utilización de forma libre y gratuita (con las limitaciones que más adelante se detallan) por todos los usuarios del repositorio y del portal e-ciencia, el autor CEDE a la Universidad Pontificia Comillas de forma gratuita y no exclusiva, por el máximo plazo legal y con ámbito universal, los derechos de digitalización, de archivo, de reproducción, de distribución, de comunicación pública, incluido el derecho de puesta a disposición electrónica, tal y como se describen en la Ley de Propiedad Intelectual. El derecho de transformación se cede a los únicos efectos de lo dispuesto en la letra (a) del apartado siguiente.

3º. Condiciones de la cesión.

Sin perjuicio de la titularidad de la obra, que sigue correspondiendo a su autor, la cesión de derechos contemplada en esta licencia, el repositorio institucional podrá:

(a) Transformarla para adaptarla a cualquier tecnología susceptible de incorporarla a internet; realizar adaptaciones para hacer posible la utilización de la obra en formatos electrónicos, así como incorporar metadatos para realizar el registro de la obra e incorporar "marcas de agua" o cualquier otro sistema de seguridad o de protección.

[1] Especificar si es una tesis doctoral, proyecto fin de carrera, proyecto fin de Máster o cualquier otro trabajo que deba ser objeto de evaluación académica.

(b) Reproducir la en un soporte digital para su incorporación a una base de datos electrónica, incluyendo el derecho de reproducir y almacenar la obra en servidores, a los efectos de garantizar su seguridad, conservación y preservar el formato.

(c) Comunicarla y ponerla a disposición del público a través de un archivo abierto institucional, accesible de modo libre y gratuito a través de internet.²

(d) Distribuir copias electrónicas de la obra a los usuarios en un soporte digital.³

4º. Derechos del autor.

El autor, en tanto que titular de una obra que cede con carácter no exclusivo a la Universidad por medio de su registro en el Repositorio Institucional tiene derecho a:

a) A que la Universidad identifique claramente su nombre como el autor o propietario de los derechos del documento.

b) Comunicar y dar publicidad a la obra en la versión que ceda y en otras posteriores a través de cualquier medio.

c) Solicitar la retirada de la obra del repositorio por causa justificada. A tal fin deberá ponerse en contacto con el vicerrector/a de investigación (curiarte@rec.upcomillas.es).

d) Autorizar expresamente a COMILLAS para, en su caso, realizar los trámites necesarios para la obtención del ISBN.

e) Recibir notificación fehaciente de cualquier reclamación que puedan formular terceras personas en relación con la obra y, en particular, de reclamaciones relativas a los derechos de propiedad intelectual sobre ella.

5º. Deberes del autor.

El autor se compromete a:

a) Garantizar que el compromiso que adquiere mediante el presente escrito no infringe ningún derecho de terceros, ya sean de propiedad industrial, intelectual o cualquier otro.

[2] En el supuesto de que el autor opte por el acceso restringido, este apartado quedaría redactado en los siguientes términos:

(c) Comunicarla y ponerla a disposición del público a través de un archivo institucional, accesible de modo restringido, en los términos previstos en el Reglamento del Repositorio Institucional

[3] En el supuesto de que el autor opte por el acceso restringido, este apartado quedaría eliminado.

b) Garantizar que el contenido de las obras no atenta contra los derechos al honor, a la intimidad y a la imagen de terceros.

c) Asumir toda reclamación o responsabilidad, incluyendo las indemnizaciones por daños, que pudieran ejercitarse contra la Universidad por terceros que vieran infringidos sus derechos e intereses a causa de la cesión.

d) Asumir la responsabilidad en el caso de que las instituciones fueran condenadas por infracción de derechos derivada de las obras objeto de la cesión.

6º. Fines y funcionamiento del Repositorio Institucional.

La obra se pondrá a disposición de los usuarios para que hagan de ella un uso justo y respetuoso con los derechos del autor, según lo permitido por la legislación aplicable, y con fines de estudio, investigación, o cualquier otro fin lícito. Con dicha finalidad, la Universidad asume los siguientes deberes y se reserva las siguientes facultades:

a) Deberes del repositorio Institucional:

- La Universidad informará a los usuarios del archivo sobre los usos permitidos, y no garantiza ni asume responsabilidad alguna por otras formas en que los usuarios hagan un uso posterior de las obras no conforme con la legislación vigente. El uso posterior, más allá de la copia privada, requerirá que se cite la fuente y se reconozca la autoría, que no se obtenga beneficio comercial, y que no se realicen obras derivadas.

- La Universidad no revisará el contenido de las obras, que en todo caso permanecerá bajo la responsabilidad exclusiva del autor y no estará obligada a ejercitar acciones legales en nombre del autor en el supuesto de infracciones a derechos de propiedad intelectual derivados del depósito y archivo de las obras. El autor renuncia a cualquier reclamación frente a la Universidad por las formas no ajustadas a la legislación vigente en que los usuarios hagan uso de las obras.

- La Universidad adoptará las medidas necesarias para la preservación de la obra en un futuro.

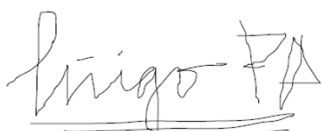
b) Derechos que se reserva el Repositorio institucional respecto de las obras en él registradas:

- Retirar la obra, previa notificación al autor, en supuestos suficientemente justificados, o en caso de reclamaciones de terceros.

Madrid, a 22. de julio de 2014.

ACEPTA

Fdo

A handwritten signature in black ink, appearing to read 'Inigo FA', written over a horizontal line.

Proyecto realizado por el alumno:

Íñigo Fernández de Araoz García - Miñaur

Fdo: Fecha:/...../.....

Autorizada la entrega del proyecto cuya información no es de
carácter confidencial

LOS DIRECTORES DEL PROYECTO:

Prof. Dr. Antonio Muñoz San Roque

Fdo: Fecha:/...../.....

Prof. Dr. José Villar Collado

Fdo: Fecha:/...../.....

Vº Bº DEL COORDINADOR DE PROYECTOS

Prof. Dr. Álvaro Sánchez Miralles

Fdo: Fecha:/...../.....

2013/14

Universidad Pontificia Comillas

Escuela Superior de Ingeniería Comillas (ICAI)

Memoria descriptiva del proyecto fin de grado en
Ingeniería Electromecánica - electrónica



DESARROLLO DE UN PROTOTIPO DE ORDENADOR PARA BICICLETA DE MONTAÑA

DIRIGIDO POR: ANTONIO MUÑOZ SAN ROQUE & JOSÉ VILLAR COLLADO

IÑIGO FERNÁNDEZ DE ARAOZ GARCÍA - MIÑAUR

RESUMEN

1. INTRODUCCIÓN

La evolución de los teléfonos móviles en los últimos años ha mostrado unos ratios de penetración y crecimiento muy superiores a los que experimentaron otras tecnologías disruptivas y otros productos innovadores de gran consumo. Los smartphones ya son mini ordenadores que incluyen bibliotecas de música, potentes cámaras, GPS con mapas detallados, agendas con completas funcionalidades, etc.

Muchas de sus aplicaciones se han orientado a la práctica y entrenamiento deportivo utilizando capacidades tales como la conexión 3G o 4G a servidores de mapas, el sistema de localización GPS, y la conexión *bluetooth* opcional con dispositivos de medida de cadencia y frecuencia cardíaca.

Sin embargo, a pesar de sus indudables capacidades, los retos relativos a la duración de la batería, la cobertura y la robustez del dispositivo siguen buscando sólidas soluciones.

Se desea por ello diseñar y construir un dispositivo de bajo coste y bajo consumo, que recibiendo la localización a través de GPS y permitiendo la recepción de sensores de la bicicleta o de medida de la frecuencia cardíaca del ciclista, sirva para mejorar un entrenamiento deportivo. Para ello el dispositivo tendrá que ser capaz de precargar rutas con la peculiaridad de asociar unos objetivos de pulsaciones para el ejercicio. Durante el transcurso de la ruta, se deberá monitorizar desde una pantalla la velocidad y la posición, además de facilitar medidas de tiempo.

2. OBJETIVOS

El objetivo final de este proyecto es el desarrollo de un sistema de ordenador de a bordo para bicicletas de montaña que cumpla los siguientes requisitos:

a) El sistema ha de ser capaz de cargar un fichero con el recorrido de la ruta en el formato estándar GPX.

b) Permitir programar el perfil de frecuencia cardíaca que se desea para el entrenamiento.

c) El equipo dispondrá de una interfaz para la monitorización de la actividad, en la que durante la marcha, se muestre la siguiente información:

- Posicionamiento GPS y localización respecto a ruta.

- Medidas sobre el estado físico del deportista; plan programado de pulsaciones y los valores reales según se obtienen.
- Medida en tiempo real de la distancia recorrida, la velocidad de cruce, evolución de altitud de la ruta y tiempo de ejercicio.

Además, el proyecto utiliza software libre, de forma que sea posible para otros usuarios ampliar el equipo con nuevos sensores e interfaces.

3. ARQUITECTURA

El sistema desarrollado se compone de los elementos que se muestran en la Figura 1.

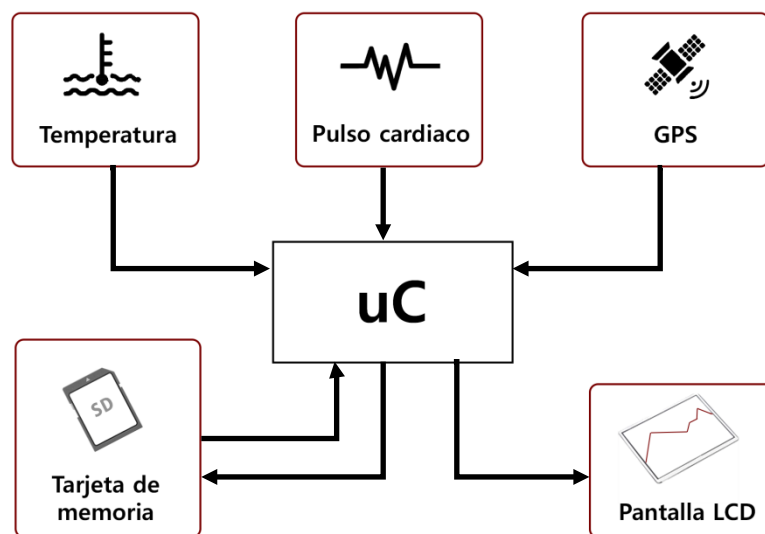


FIGURA 1. COMPONENTES DEL PROTOTIPO

El microcontrolador de bajo coste Arduino es el encargado de sincronizar las medidas obtenidas por los periféricos de la imagen. Estos hacen posible obtener la información en el transcurso del recorrido:

- GPS: Aunque la actualización de la localización no se haga internamente en una interrupción, se utiliza el timer0 del Arduino para administrar el periodo de solicitud de la posición.
- Pantalla y cronómetro: La pantalla se actualiza mediante comunicación serie con el bus I²C. Internamente actualiza las medidas de tiempo de la ruta, administradas con el timer1. El microcontrolador gestiona el tiempo con precisión de milisegundos, en cambio, la pantalla tarda un segundo en actualizarse. La alimentación de la pantalla y sus entradas de datos trabajan a 3.3V, por lo que ha sido necesario utilizar un divisor de resistencias para reducir los niveles de tensión desde el Arduino Mega a la pantalla.

- ▶ Tarjeta SD: La comunicación con la tarjeta de memoria se realiza a través del estándar SPI. El Arduino posee unos pines específicos para este tipo de comunicación, haciendo posible la lectura y escritura de archivos.
- ▶ Sensor de temperatura: La lectura se hace de manera digital. Recibe información periódicamente y después de transformar el dato a grados Celsius, se expone en pantalla. La señal se transforma a digital en el propio sensor.
- ▶ Sensor de pulso cardíaco: Cada 2ms el timer2 del Arduino actualiza el valor de lectura.

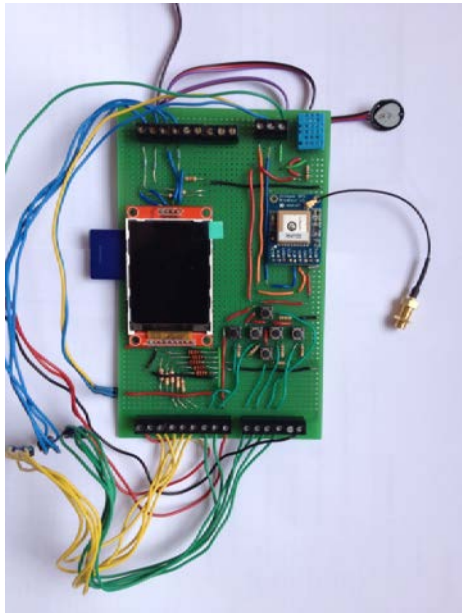


FIGURA 2. PROTOTIPO

Tras 250 muestras de este tipo, el programa detecta los picos y calcula el número de pulsaciones en ese tiempo. La medida que se expone en pantalla es la extrapolación de ese dato a un minuto. La señal analógica se transforma a digital internamente en el sensor.

▶ Pulsadores: La interfaz se gestiona con seis pulsadores 'pull-down'. Cada uno se lee de manera digital con un detector de flanco.

La ruta que se carga en la tarjeta de memoria debe prepararse

en un formato reducido que recoge exclusivamente los valores útiles para esta aplicación. Esta misión la realiza un programa en el ordenador, que reúne las siguientes funcionalidades:

- ▶ Cargar ruta en formato GPX
- ▶ Presentar en una gráfica el perfil de altitud
- ▶ Mostrar en una ventana, la información de cada punto definido en la ruta.
- ▶ Asociar con un cursor el entrenamiento cardíaco a cada punto de la ruta.
- ▶ Exportar los valores en un formato descifrable por el prototipo.

La interacción del programa que prepara la ruta y el dispositivo que la monitoriza en su transcurso compone el conjunto del proyecto.

4. RESULTADOS

Tras las pruebas en campo, con rutas de distintos tipos se han obtenido resultados satisfactorios. La operabilidad de este dispositivo es suficientemente potente para deportistas centrados en el ciclismo. Su bajo consumo hace posible trabajar más de 8 horas a pleno rendimiento y su precisión permite extraer valores para su posterior análisis (error de posición

inferior a los 10m). En la Figura 3 se compara el resultado de precisión de la ruta programada (en línea roja) y los puntos por los que el usuario pasa y recoge la posición.

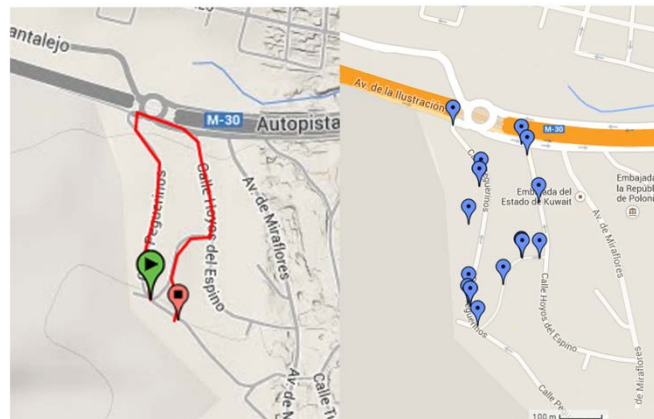


FIGURA 3. COMPARACIÓN RUTAS EN PRUEBA

Se ha llegado a realizar un primer prototipo de ordenador de a bordo. Capaz de operar en dos modos: con o sin ruta precargada. Puede interactuar con información escogida por el usuario y fácilmente insertable con una tarjeta de memoria. A través de seis sencillos pulsadores, el usuario gestiona la información que desea mostrar sobre la ruta.

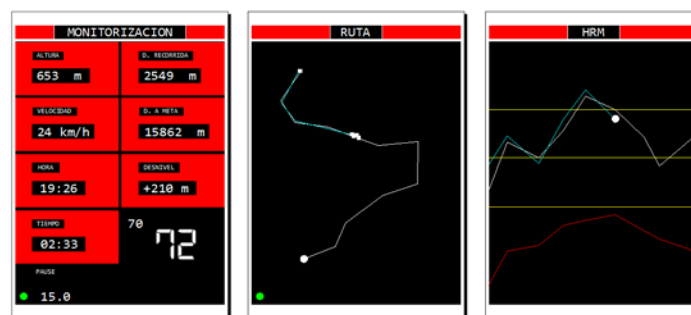


FIGURA 4. PANTALLAS PRINCIPALES

Se puede escoger entre las tres pantallas, que muestra la Figura 4. En la pantalla de monitorización se presentan valores numéricos sobre el estado del entrenamiento. Como son, la distancia recorrida o el desnivel acumulado. Desde aquí se puede acceder a la tercera pantalla que expone el perfil de altitudes y la variación en las pulsaciones del corredor. En la segunda, se muestra la ruta a la que se superpone el tramo ya realizado.

Uno de los resultados obtenidos ha sido el programa que gestiona una ruta estándar en formato GPX. Interpreta la sintaxis de este archivo y consigue extraer la información contenida. El valor diferenciador, lo añade la posibilidad de poder personalizar las rutas con un plan de entrenamiento cardiaco. Con la misma interfaz de la Figura 5, se consigue cumplir ambos objetivos.

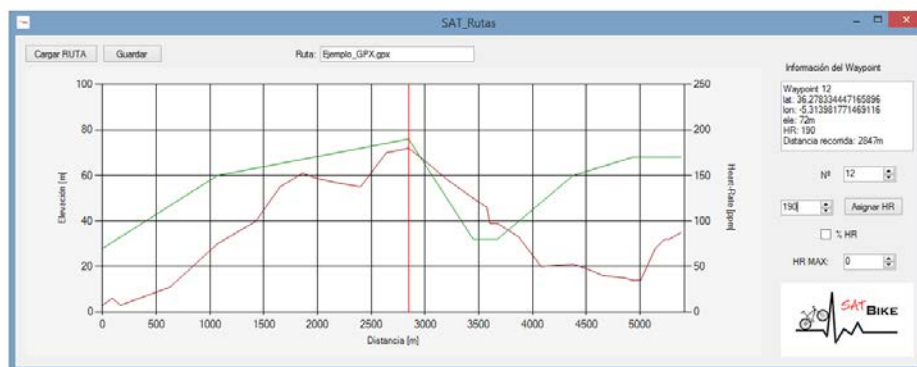


FIGURA 5. INTERFAZ SAT_RUTAS

5. CONCLUSIONES

Pese a la infinidad de aplicaciones de móvil que ofrecen los servicios aquí descritos, los deportistas de hoy quieren aparatos exclusivos, más personalizables, de menor consumo, más ergonómicos y con mayor precisión. En este proyecto se ha realizado un prototipo que con la personalización de rutas añade valor a las ciclocomputadoras corrientes. Aún siendo un primer prototipo, ofrece un servicio completo aunque abierto a múltiples mejoras, que consigan perfeccionar un producto. Un producto, que pueda responder a las diferentes demandas de los consumidores, en bicicletas de montaña o ciclismo en carretera.

Las empresas tecnológicas desarrollan constantemente nuevas funcionalidades, por los diferentes gustos del consumidor o por la diferenciación de cada deporte. Crear un prototipo escalable y moldeable para cada deportista parece interesante y que seguro tendría cabida en un mercado tan competido. Con este dispositivo se han sentado las bases para poder personalizar los entrenamientos y recortar segundos.

6. BIBLIOGRAFÍA

- [1] C. Cupeiro Durán, J. López González, D. Redondo Arroyo, "Ordenador de a bordo (visualización de velocidad, distancia, temperatura...)", Universidad Rey Juan Carlos, 2012
- [2] Ron/Spinning magnets - Electricbike "Qoros eBIQE, Exotic Concept E-Bike", Mar. 2014
- [3] Z. Romano, "DIY bicycle computer with Arduino", Ago. 2013
- [4] E. Terol, - Ecoavant "La bicicleta conquista Europa", Oct. 2013
- [5] J. Leber - "A Shrinking Garmin Navigates the Smartphone Storm" Mar.2013

ABSTRACT

1. INTRODUCTION

The evolution of mobile phones in recent years, has shown penetration and growth ratios higher than those experienced by other disruptive technologies. Smartphones today are mini computers, with incredible libraries that include music, powerful cameras, GPS with detailed maps, agendas with full functionalities, etc.

Many applications have been aimed to the sports training and competing. Use facilities such as 3G or 4G connection to maps servers, GPS location technologies, and optional *bluetooth* connection to devices that measure cadence and heart rate.

However, despite its undoubted capabilities, challenges related to battery life, broadband coverage and robustness of the device are still looking for solid solutions.

The idea is to design and build a low cost device working on low power consumption mode. Able to receive the location through GPS and other information related with the bike and sportsmen throughout sensors. This project will help complete a training performance. The device must be able to preload customized routes with heart pulse objectives for the track. Monitor in the screen speed, position and time measures, will be necessary too.

2. OBJECTIVES

The ultimate goal of this project is to develop an on-board computer system for mountain bikes that meets the following requirements:

- a) The system must be able to upload a file with the path of the route in the standard GPX format.
- b) Customize the user's desired heart rate training profile.
- c) The device will have an interface for monitoring activity, where during exercise, the following information is displayed:
 - GPS positioning and location in a route.
 - Measures on the physical condition of the athlete, programmed pulse rate plan and actual values obtained.
 - Real-time measurement of distance, cruise speed, altitude evolution of the route and time .

In addition, the project is built on a free software basis, so that every owner can expand the system with new sensors and original interfaces.

3. ARCHITECTURE

The system has been developed with the elements shown in Figure 1.

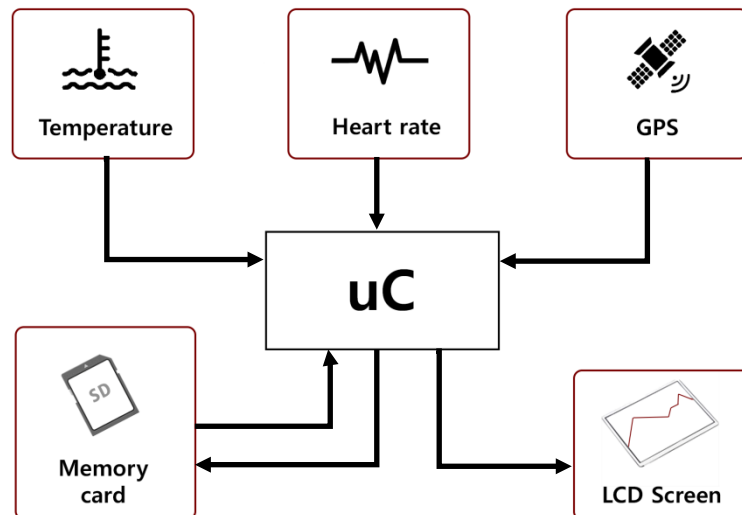


FIGURE 1. PROTOTYPE COMPONENTS

The low cost Arduino microcontroller is responsible for the synchronization of the measurements obtained by the peripherals shown above. The information during travel is obtained due to:

- ▶ GPS: Although it is not controlled directly with interruptions, timer0 is used to compare time while a position request takes place.
- ▶ Screen and chronometer: The display is updated through serial communication with the bus I²C. Internally, time measurements are obtained with Arduino's timer1. The microcontroller manages the time with microseconds precision, while each second changes the new value is sent to the display. The display power and data signals operate at 3.3V. A voltage divider with resistors has been implemented to reduce the logic levels from the Arduino Mega to the screen.
- ▶ SD card: Communication with the memory card is performed via the SPI standard. The Arduino has specific pins for this type of communication. These libraries enable reading and writing to files.

- Temperature sensor: The reading is done digitally. The information is received periodically and after transforming the input to degrees Celsius it is displayed on the LCD screen.
- Heart rate sensor: The timer2 inside the Arduino stores a value every 2ms. After 250 samples, an algorithm detects every peak and obtains the number of pulses in that time. The value shown is an extrapolation of the last result to one minute.

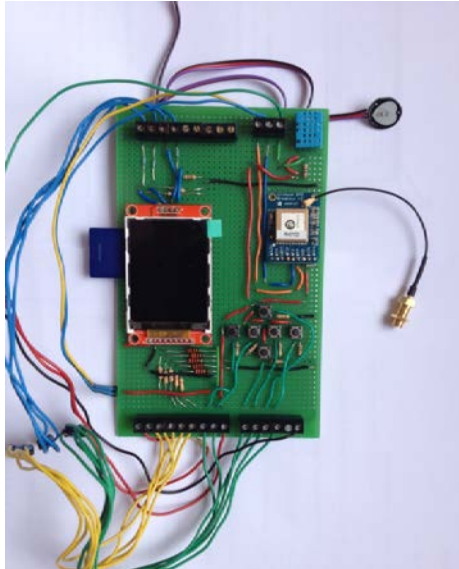


FIGURE 2. PROTOTYPE

► Buttons: The interface is managed with six 'pull-down' buttons. Each is read digitally through an edge detector.

The route loaded into the memory card should be prepared in a specific format that collects only useful values for this application. This task is performed by a computer

program (encountered in the objectives of the project) which includes the following features:

- Load route in GPX extension
- Exhibit a graph with the altitude variation
- Displayed in a window, the information for each waypoint on the trail.
- Link with a cursor the expected values of heart rate
- Export values in a legible format by the prototype

The interaction of these two individual systems complete the entire project.

4. RESULTS

After successful field tests with different types of trails, the operability of the device is powerful enough for big sport fans, mainly focused on cycling. Its low power consumption makes it possible to work more than 8 hours at full capacity and the accuracy can extract values for further analysis (position error less than 10m). In Figure 3 the results of a test are shown, designed route on red and the track made by points in blue, the values obtained during the performance.

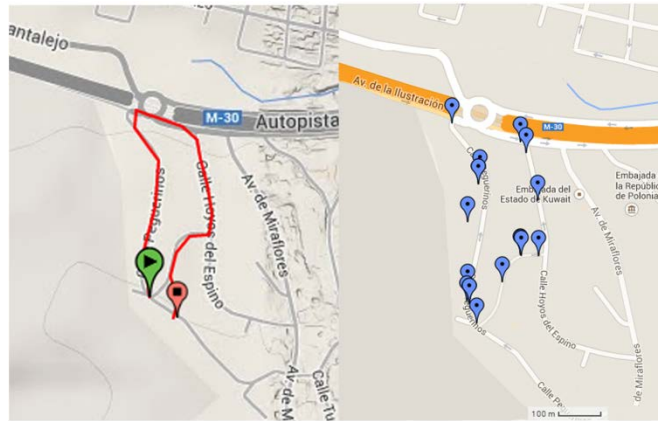


FIGURE 3. TRACKS COMPARED

It has come to carry out a first on-board computer prototype. Able to operate in two modes: with or without preloaded route. Interacting with information selected by the user inserting memory card. Through six simple buttons, the user manages to display information about the path.



FIGURE 4. MAIN SCREENS

The user can choose between three screens as shown in Figure 4. Numerical values are shown in the monitoring screen. Such as, the distance covered or the accumulated height difference. From here you can access the third screen, which sets the altitude graph and the heart rate variation of the runner. In the second, a drawing with the appearance of the path and the trail that the sportsmen is performing.

One of the partial results is the program that transforms the incoming route in GPS extension. Analyze the syntax of this file and extract the information obtained is one of the goals accomplished. In addition to customizing the routes with a workout plan. All this thanks, to a visual interface that increases the value of this application.

One of the partial results was the program that manages a standard route GPX. Analyzes the syntax of this file and extract the information contained. The differentiating value, is being able to customize routes in a cardio workout plan. With the same interface shown in Figure 5, both objectives are achieved.

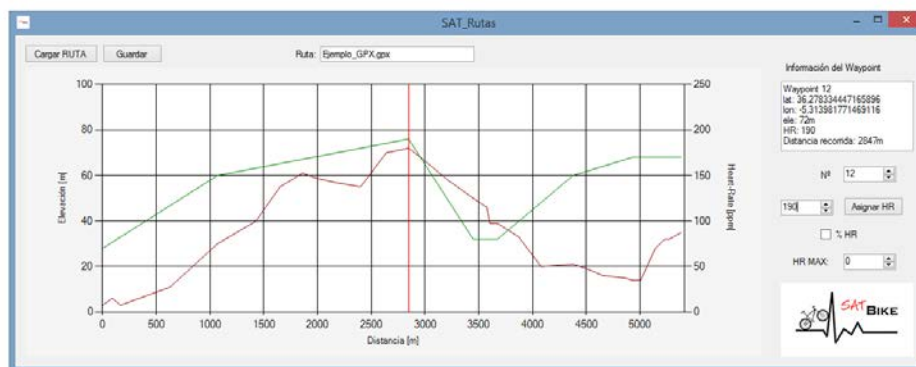


FIGURE 5. SAT_RUTAS INTERFACE

5. CONCLUSIONS

Despite the myriad of mobile applications that provide the services described here, elite athletes want exclusive devices, with low power consumption, ergonomic designs and precise reception. This project has added value to the current cycle computers by being able to customize the track based on cardio workouts. Even as a first prototype, offers full service, but is open to multiple improvements to grab a refine product. A product that can meet the different demands of consumers, mountain biking or road cycling.

Technology companies constantly develop new functionalities for different consumer tastes or differentiation of each sport. Create a scalable and moldable prototype to every athlete looks interesting and surely would have a place in a highly competitive market. With this device everyone could customize workouts and save seconds.

6. BIBLIOGRAPHY

- [1] C. Cupeiro Durán, J. López González, D. Redondo Arroyo, "Ordenador de a bordo (visualización de velocidad, distancia, temperatura...)", Universidad Rey Juan Carlos, 2012
- [2] Ron/Spinning magnets - Electricbike "Qoros eBIQE, Exotic Concept E-Bike", Mar. 2014
- [3] Z. Romano, "DIY bicycle computer with Arduino", Ago. 2013
- [4] E. Terol, - Ecoavant "La bicicleta conquista Europa", Oct. 2013
- [5] J. Leber - "A Shrinking Garmin Navigates the Smartphone Storm" Mar.2013

AGRADECIMIENTOS

En primer lugar quiero dar las gracias a mis directores de proyecto, Antonio y José. Su paciencia y el tiempo dedicado ha sido clave para llegar a terminar la memoria. También agradecer a la Universidad Pontificia Comillas por su atención durante estos cuatro años. La profesionalidad de sus profesores y el talento de sus alumnos, ha hecho que me encuentre cómodo en un ambiente donde aprender ha sido la principal tarea. Mencionar a profesores con un cariño especial como a Ángela Jiménez Casas que me ha apoyado en cada decisión de la vida. Otros como Juan Luis Zamora, Félix Alonso o Álvaro Sánchez me han enseñado lo que es el compromiso del trabajo bien hecho.

Quisiera nombrar aquí a mis compañeros de ICAI, con ellos ha sido posible desarrollarse en un ambiente cargado de compañerismo. Una larga lista de nombres y momentos que han hecho posible escribir esto hoy. Mis amigos de toda la vida que se han interesado por el estado de SATBike en cada momento, animándome a seguir con ganas. Un aperitivo o un partido de fútbol forma parte también del saber desconectar para volver a crear con nuevas ideas.

Mi familia es el eje principal que me lleva apoyando en todo lo que hago. Desde mis abuelos hasta mis sobrinos. Gracias a todos. En especial a mis padres y a mis tres hermanos Lucas, Carmen y María, son lo más importante que tengo. Ali, gracias a ti también, cada día eres más familia, gracias por tu apoyo y estar dispuesta a soportar a un futuro ingeniero con todo lo que eso conlleva.

Para acabar querría citar a todos aquellos que forman parte de mi vida, que merecen una distinción especial. Por las buenas amistades que tengo y por lo contento y orgulloso que estoy de haber realizado este proyecto.

A mis padres, por su
cariño, su paciencia y su
sacrificio en cada uno de
estos años.

I. MEMORIA	9
1. INTRODUCCIÓN	9
1.1 ESTUDIO DE LOS TRABAJOS EXISTENTES / TECNOLOGÍAS EXISTENTES	9
1.2 MOTIVACIÓN DEL PROYECTO	11
1.3 OBJETIVOS DEL PROYECTO	12
1.4 PLAN DE TRABAJO	12
1.5 RECURSOS / HERRAMIENTAS EMPLEADAS	13
2. ARQUITECTURA DEL SISTEMA	15
2.1 MICROCONTROLADOR ARDUINO	15
2.2 PANTALLA LCD + GESTIÓN TARJETA SD	17
2.3 RECEPTOR GPS	18
2.4 SENSOR DE FRECUENCIA CARDIACA	19
2.5 SENSOR DE TEMPERATURA	21
2.6 MANDOS - BOTONES	21
2.7 SOPORTE	22
3. SATBIKE	25
3.1 DIAGRAMA DE ESTADOS	25
4. SAT_RUTAS	29
4.1 FORMATO GPX	29
4.1 ARCHIVO SALIDA	30
5. RESULTADOS	31
6. CONCLUSIONES	33
7. FUTUROS DESARROLLOS	35
7. 1 MODO INDIVIDUAL	35
7. 2 MODO GRUPO	35
8. REFERENCIAS	37
II. PLANOS	39

III. PRESUPUESTO	43
1. MEDICIONES	43
1.1 COMPONENTES PRINCIPALES	43
1.2 EQUIPO Y HERRAMIENTAS	44
1.3 SOFTWARE	44
1.4 MANO DE OBRA DIRECTA	44
2. PRECIOS UNITARIOS	45
2.1 COMPONENTES PRINCIPALES	45
2.2 EQUIPO Y HERRAMIENTAS	45
2.3 SOFTWARE	46
2.4 MANO DE OBRA DIRECTA	46
3. SUMAS PARCIALES	47
3.1 COMPONENTES PRINCIPALES	47
3.2 EQUIPO Y HERRAMIENTAS	47
3.3 SOFTWARE	48
3.4 MANO DE OBRA DIRECTA	48
4. PRESUPUESTO GENERAL	49
IV. MANUAL DE USUARIO	51
1. GESTIÓN DE RUTA	51
1.1 INSTALACIÓN	51
1.2 INTERFAZ	51
1.3 FUNCIONAMIENTO	52
2. EMPLEO DE SATBIKE	53
2.1 PANTALLA INICIAL	53
2.2 PANTALLA RUTA	53
2.3 PANTALLA MONITORIZACIÓN (MON)	53
2.4 PANTALLA HRM	54
2.5 PANTALLA MONITORIZACIÓN SIN RUTA (MON_SR)	54
V. CÓDIGO	55

1.SATBIKE.INO	55
2. SAT_RUTAS.VB	74
VI. HOJAS DE CARACTERÍSTICAS Y LIBRERÍAS	81
1. HOJAS DE CARACTERÍSTICAS	81
2. LIBRERÍAS	81

ÍNDICE DE FIGURAS

<i>Figura 1. Ordenador de a bordo.....</i>	<i>10</i>
<i>Figura 2. Componentes del prototipo</i>	<i>15</i>
<i>Figura 3. Imagen del prototipo.....</i>	<i>17</i>
<i>Figura 4. Interfaz de comunicación ANT.....</i>	<i>20</i>
<i>Figura 5. Señal pulso cardiaco.....</i>	<i>20</i>
<i>Figura 6. Esquema sensor pulsaciones/min</i>	<i>21</i>
<i>Figura 7. Pulsadores Prototipo.....</i>	<i>22</i>
<i>Figura 8. Pull down pulsadores.....</i>	<i>22</i>
<i>Figura 9. Detalle Explosionado Prototipo.....</i>	<i>23</i>
<i>Figura 10. Diagrama de estados</i>	<i>25</i>
<i>Figura 11. Pantalla modo libre</i>	<i>26</i>
<i>Figura 12. Mapa funcionalidad rutas.....</i>	<i>27</i>
<i>Figura 13. Inicio.....</i>	<i>27</i>
<i>Figura 14. RUTA.....</i>	<i>27</i>
<i>Figura 15. Monitorización.....</i>	<i>28</i>
<i>Figura 16. HRM.....</i>	<i>28</i>
<i>Figura 17. Transcripción SAT_RUTAS.....</i>	<i>29</i>
<i>Figura 18. Interfaz SAT_RUTAS.....</i>	<i>30</i>
<i>Figura 19. Comparación rutas prueba.....</i>	<i>31</i>
<i>Figura 20. Pantallas principales.....</i>	<i>32</i>
<i>Figura 21. Simulación 3D del prototipo</i>	<i>33</i>
<i>Figura 22. Partes interfaz SAT_RUTAS.....</i>	<i>51</i>
<i>Figura 23. Inicial + pulsadores</i>	<i>53</i>
<i>Figura 24. Ruta + pulsadores</i>	<i>53</i>
<i>Figura 25. Monitorización + pulsadores.....</i>	<i>53</i>
<i>Figura 26. HRM + PULSADORES.....</i>	<i>54</i>
<i>Figura 27. Monitorizacion sin Ruta + pulsadores.....</i>	<i>54</i>
<i>Figura 28. Diseño interfaz SAT_RUTAS.....</i>	<i>74</i>

ÍNDICE DE TABLAS

<i>Tabla 1. Cronograma.....</i>	<i>13</i>
<i>Tabla 2. Comparacion de μC.....</i>	<i>16</i>
<i>Tabla 3. Gestión EEPROM.....</i>	<i>17</i>
<i>Tabla 4. Información obtenida en GPGGA.....</i>	<i>18</i>
<i>Tabla 5. Contenido formato GPX.....</i>	<i>29</i>
<i>Tabla 6. SATB.txt.....</i>	<i>30</i>
<i>Tabla 7. Relacion de componentes.....</i>	<i>43</i>
<i>Tabla 8. Relación de equipos y herramientas.....</i>	<i>44</i>
<i>Tabla 9. Relación de herramientas sw.....</i>	<i>44</i>
<i>Tabla 10. Relación de horas de mano de obra.....</i>	<i>44</i>
<i>Tabla 11. Precio unitario componentes principales.....</i>	<i>45</i>
<i>Tabla 12. Precio unitario Equipos y herramientas.....</i>	<i>45</i>
<i>Tabla 13. Precio licencias sw.....</i>	<i>46</i>
<i>Tabla 14. Precio unitario mano de obra.....</i>	<i>46</i>
<i>Tabla 15. Sumas parciales componentes principales.....</i>	<i>47</i>
<i>Tabla 16. Sumas parciales equipos y herramientas.....</i>	<i>48</i>
<i>Tabla 17. Sumas parciales sw.....</i>	<i>48</i>
<i>Tabla 18. Sumas parciales mano de obra directa.....</i>	<i>48</i>
<i>Tabla 19. Presupuesto general.....</i>	<i>49</i>

LISTA DE ABREVIATURAS

BPM	BEATS PER MINUTE
CS	CHIP SELECT
EEPROM	ELECTRICALLY ERASABLE PROGRAMMABLE READ - ONLY MEMORY
FDM	FUSED DEPOSITION MODELING
GMT	GREENWICH MEAN TIME
GPS	GLOBAL POSITIONING SYSTEM
GPX	GPS EXCHANGE FORMAT
HRM	HEART RATE MONITOR
I ² C	INTER-INTEGRATED CIRCUIT
ICT	INFORMATION AND COMMUNICATIONS TECHNOLOGY
IDE	INTEGRATED DEVELOPMENT ENVIRONMENT
LCD	LIQUID CRYSTAL DISPLAY
LED	LIGHT EMITTING DIODE
MISO	MASTER INPUT SLAVE OUTPUT
MOSI	MASTER OUTPUT SERIAL INPUT
NMEA	NATIONAL MARINE ELECTRONICS ASSOCIATION
RS / RST	RESET
SCK / SCL	SERIAL CLOCK
SD	SECURE DIGITAL
SDA	SERIAL DATA
SPI	SERIAL PRIPHERAL INTERFACE
SRAM	STATIC RANDOM ACCESS MEMORY
TFT	THIN-FILM TRANSISTOR
USB	UNIVERSAL SERIAL BUS

I. MEMORIA

1. INTRODUCCIÓN

La evolución de los teléfonos móviles en los últimos años ha mostrado unos ratios de penetración y crecimiento muy superiores a los que experimentaron otras tecnologías disruptivas y otros productos innovadores de gran consumo. Los smartphones ya son mini ordenadores que incluyen bibliotecas de música, potentes cámaras, GPS con mapas detallados, agendas con completas funcionalidades, etc.

Muchas de sus aplicaciones se han orientado a la práctica y entrenamiento deportivo utilizando capacidades tales como la conexión 3G o 4G a servidores de mapas, el sistema de localización GPS, y la conexión *bluetooth* opcional con dispositivos de medida de cadencia y frecuencia cardiaca.

Sin embargo, a pesar de sus indudables capacidades, los retos relativos a la duración de la batería, la cobertura y la robustez del dispositivo siguen buscando sólidas soluciones.

Se desea por ello diseñar y construir un dispositivo de bajo coste y bajo consumo, que recibiendo la localización a través de GPS y permitiendo la recepción de sensores de la bicicleta o de medida de la frecuencia cardiaca del ciclista, sirva para mejorar un entrenamiento deportivo. Para ello el dispositivo tendrá que ser capaz de precargar rutas con la peculiaridad de asociar unos objetivos de pulsaciones para el ejercicio. Durante el transcurso de la ruta, se deberá monitorizar desde una pantalla la velocidad y la posición, además de facilitar medidas de tiempo.

1.1 ESTUDIO DE LOS TRABAJOS EXISTENTES / TECNOLOGÍAS EXISTENTES

Para el diseño del dispositivo descrito en el apartado anterior es necesario escoger un microprocesador, un dispositivo gps, una pantalla y un sensor receptor de frecuencia cardiaca.

Para la gestión del ordenador de a bordo es necesario un microcontrolador. Las operaciones necesarias no son excesivamente potentes y por eso un chip que trabaje con 8-bit es suficiente. Existen varios modelos de distintas marcas: los modelos dsPIC de Microchip, distintos tipos de Arduino (UNO y MEGA) y controladores de baja potencia de Texas Instruments como el MSP430.

Pese a no ser el más potente ni el más barato. La opción que mejor se adapta a las necesidades del proyecto es el Arduino, en concreto la versión MEGA. Por su carácter de software libre y su uso en trabajos de este tipo. Está compuesto por un chip ATmega 2560 que trabaja a 16MHz y el entorno de programación es sencillo. La marca Arduino ha conseguido

vender más de 500.000 dispositivos oficiales por su robustez y fácil adaptación a distintos campos.

Los dispositivos GPS cada vez son más frecuentes: en el coche, en el móvil, en un reloj, hasta en perros para no perderlos. Por poco dinero se consigue un dispositivo con precisión menor de 10m. Es interesante conocer tu posición pero sobre todo el GPS será útil para conocer con exactitud tu velocidad y distancia recorrida.

En cambio, un diseño reducido (del tamaño de un reloj) únicamente ha llegado a marcas de renombre como Garmin, Polar, Suunto o Tomtom. Son dispositivos que rondan los 300€. Como el GARMIN GPS EDGE en todas sus versiones (500, 510, 800, etc), que es un equipo que sirve de inspiración para este proyecto.

Otro proyecto de referencia es el de unos estudiantes de la universidad Rey Juan Carlos con la misma plataforma de software abierto, desarrollaron un ordenador de a bordo basado en un GPS para su vehículo.

La pantalla es una pieza clave en el desarrollo del proyecto. Es la parte que interactúa con el usuario. Debe ser de sencillo manejo y capaz de exponer claramente el seguimiento del deportista. Por eso se ha escogido una pantalla que no es táctil, facilitando así las transiciones de menús. Las aplicaciones de móvil que ofrecen servicios para el uso deportivo como este trabajo de la Figura 1 son una referencia de interacción.



FIGURA 1. ORDENADOR DE A BORDO

OruxMaps o Androzic son aplicaciones para Android en las que es posible ver mapas de rutas gestionadas y obtener información como la distancia recorrida, velocidad media, dirección de la ruta, etc. Otras en cambio como Endomondo o SportsTracker han añadido una comunidad de deporte, con perfiles para compartir el ejercicio realizado, publicar tus tiempos y compartir rutas. Cada vez son más frecuentes las recomendaciones de rutas por senderos o por campo a través. Servidores como Wikiloc ponen en contacto a personas para compartir sus itinerarios. Muchos de los dispositivos de las marcas antes citadas son capaces de esto, crear y cargar rutas propias o ajenas e ir monitorizando información según se realizan los itinerarios.

No existen proyectos como éste en un ambiente universitario, en el que se desarrollen estas tecnologías como algo global. Aunque herramientas como las que se pretenden usar han sido estudiadas, existen y con logros fascinantes.

Con la salud como objetivo se han desarrollado muchos dispositivos capaces de medir frecuencia cardiaca y muchos otros parámetros. Las conexiones entre relojes y sensores para la práctica deportiva suele ser mediante tramas codificadas, haciendo conexiones cerradas que facilitan la recepción en ambientes multitudinarios. Por seguir con una visión de un sistema abierto se planteó escoger la opción de protocolo ANT+. Un sistema de transmisión que opera en 2,4GHz y que las mejores firmas han incluido en sus dispositivos.

Existen varias aplicaciones con esta tecnología: unos estudiantes de la universidad de Malasia fueron capaces de crear una red inalámbrica en la que existía un traspaso de tramas de protocolos ANT+ en un ambiente de entrenamiento. Un corredor utilizó los mismos módulos para exponer en una camiseta repleta de LEDs el pulso de su corazón. Sin embargo, la conexión con estos equipos no ha sido posible. Por contra, se ha añadido un sensor, basado en el cambio de reflexión de luz en la piel con el flujo sanguíneo. Proyectos con este tipo de periféricos hay muchos. Uno particular que llama la atención es el de la Universidad sueca Viktoria ICT que incorpora en un casco de bicicleta este medidor con comunicación *bluetooth* que gestiona desde un dispositivo móvil.

1.2 MOTIVACIÓN DEL PROYECTO

Existen varias razones principales por las que este proyecto parece atractivo. Permite prescindir del móvil por los siguientes inconvenientes:

- En varios lugares donde es común la práctica deportiva, como en el monte no hay una cobertura estable para utilizar las aplicaciones que alberga un *smartphone*.
- El uso del teléfono para servicios de este tipo aumenta el rendimiento y hace que el consumo de la batería se dispare.
- Los sistemas son cerrados y permiten poca personalización.
- Equipos delicados con muchas características no resistentes a un ejercicio intenso en bicicleta.

La propuesta que se plantea pretende, además de paliar las necesidades anteriores, sumar:

- Personalizar el entrenamiento en rutas basándose en la evolución de frecuencia cardiaca.
- Desarrollo de un dispositivo de bajo coste
- Posibilidad de ampliar las medidas que captura el ordenador. Escalable según los usuarios.
- Libre de conexión mediante cobertura y capaz de precargar rutas sin acceso web.

1.3 OBJETIVOS DEL PROYECTO

El objetivo final de este proyecto es el desarrollo de un sistema de ordenador de a bordo para bicicletas de montaña que cumpla los siguientes requisitos:

a) El sistema ha de ser capaz de cargar un fichero con el recorrido de la ruta con la formato .gpx.

b) El sistema permitirá programar el perfil de frecuencia cardíaca que se desea para el entrenamiento.

c) El sistema dispondrá de un interfaz para la monitorización de la actividad durante la marcha que muestre la siguiente información:

- Posicionamiento GPS y localización en mapa.
- Medidas sobre el estado físico del deportista.
- Medida en tiempo real de la distancia recorrida y la velocidad de crucero.

1.4 PLAN DE TRABAJO

El proyecto se ha ido realizando desde lo más importante como la pantalla y el GPS, hasta incorporar los detalles más simples.

Uno de los puntos a favor del desarrollo de este proyecto es que con la conexión de los primeros periféricos el equipo adquiere funcionalidad. Tras esto se trata de perfeccionar cada conexión y hacerlo fácilmente accesible para el usuario. Un trabajo que se va modificando según se prueban las herramientas con el prototipo. El plan de trabajo realizado incluye las siguientes tareas:

Tarea 1: Diseño del sistema.

Diciembre 2013: Detalle y pedido de los componentes.

 Conexionado de los módulos principales.

Tarea 2: Programación.

 Unificación de código de los módulos.

 Programación de menús e instrucciones.

 Desarrollo del dibujo de mapas e lectura de archivo de extensión .gpx

Tarea 3: Implantación del circuito.

 Desarrollo de sujeción del prototipo.

Tarea 4:  Pruebas y ensayos.

Tarea 5: Documentación.

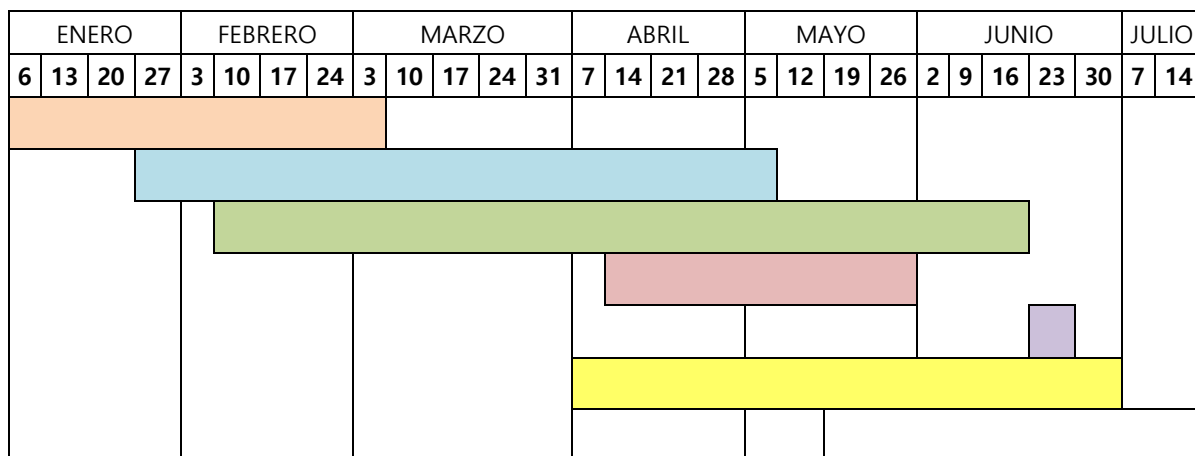


TABLA 1. CRONOGRAMA

1.5 RECURSOS / HERRAMIENTAS EMPLEADAS

Los recursos empleados en este proyecto se pueden clasificar en hardware y software.

- Hardware:
 - Arduino MEGA
 - Pantalla TFT01 - 2.2SP 240 x 320 + lector SD
 - GPS MTK3339 Adafruit
 - Antena GPS con base magnética
 - DHT11 - Sensor de Temperatura
 - Sensor ritmo cardiaco - Pulse sensor Amped
 - Componentes electrónicos (resistencias, cables, pulsadores, tarjeta, etc.)
 - Equipo de soldadura: soldador, estaño, alicates, pinza y desoldador.
- Software:
 - Arduino IDE
 - Visual Studio 2008
 - KiCad para dibujar el cableado
 - SketchUp para el desarrollo 3D del soporte

2. ARQUITECTURA DEL SISTEMA

En este capítulo se explican las partes de las que consta el prototipo. Se analizará con detalle la manera en la que son capaces de comunicarse unas con otras, como sus protocolos o estándares particulares. El prototipo se compone de los elementos que se muestran en la Figura 2.

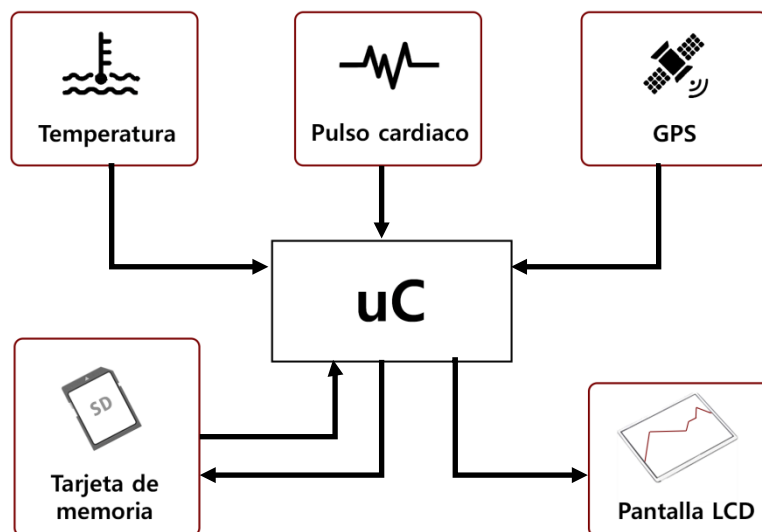


FIGURA 2. COMPONENTES DEL PROTOTIPO

2.1 MICROCONTROLADOR ARDUINO

Para el desarrollo se ha utilizado la tarjeta Arduino MEGA, que es un módulo basado en el microcontrolador ATmega2560. Se habían realizado previos proyectos de menor tamaño en otras versiones de Arduino, y los resultados habían sido satisfactorios. Su popularidad, su robustez y la filosofía 'open source' encajaban con el desarrollo del equipo que se quería llevar a cabo.

Comparado con otros equipos similares como se muestra en la Tabla 2, tiene la mayor memoria de programa. Pese a no ser el más potente del mercado, ni el más pequeño en dimensiones, durante el desarrollo se han encontrado múltiples facilidades. A la hora de utilizar librerías por ejemplo, se ha incorporado una biblioteca de funciones previamente definidas que coordinan la pantalla. Su cantidad de pines posibilitan ampliar el proyecto para que los usuarios lo puedan personalizar a su gusto.

Marca	Arduino	Microchip	Texas Instrument
Modelo	Arduino MEGA 2560 / ATM2560	dsPIC18F2550	MSP430F5418
Velocidad (MHz)	16	40	18
Timers	2 x 8-bit 4 x 16-bit	1 x 8-bit 3 x 16-bit	3 x 16-bit
Memoria programa	256kB	32kB	128kB
EEPROM	4kB	256B	512B
A/D	16 canales, 10-bit	13 canales, 10-bit	16 canales, 12-bit
Comunicación	4 x UART 1 x SPI	1 x UART 1 x SPI/I ² C	4 x Serial
Tensión de trabajo	5V	2.0V to 5.5V	2.2V to 3.6V
Entorno de trabajo	Arduino IDE	MPLAB	Energia
Precio	41,00€ / 16,72€	5,16€	5,79€

TABLA 2. COMPARACION DE μC

En relación a la transferencia de datos con los sensores, se han empleado tres timers:

- El timer0, de 8 bits se utiliza para calcular la función interna millis(), que acumula el tiempo que el programa lleva operativo. Una vez se desborda el valor, vuelve a comenzar. La comparación con el número que alberga esta operación hace posible enviar las tramas de posicionamiento. El GPS envía las tramas según el protocolo NMEA y espera contestación.
- El timer1 aloja con precisión de microsegundos el cronómetro de las rutas. Con un preescalado se reduce la velocidad del reloj (16MHz) para gestionar los segundos, minutos y horas del ejercicio.
- La llamada al sensor de frecuencia cardiaca se realiza mediante una interrupción. Cada 2ms se recoge el valor sensor. Internamente el microcontrolador hace la conversión de analógico a digital.

Los buses de comunicación utilizados son I²C y SPI. Esta última interfaz serie solamente es accesible con los pines preparados para ello (50, 51, 52 y 53). La transacción de archivos con la tarjeta SD, requería esta interfaz de comunicación dejando los otros periféricos con comunicación serie I²C.

Esta tarjeta consta de tres tipos de memoria:

- 256kB de memoria flash: Aunque 8kB contienen el gestor de arranque, el programa total ocupa 85kB, dejando memoria libre para nuevas funcionalidades.
- 8kB de memoria SRAM: Aquí residen los valores que toman las variables del programa.

- 4kB de EEPROM: Recoge los datos por los que transcurren las rutas que se quieren registrar. Esta memoria no volátil se ha ordenado según muestra la Tabla 3.

Reg DEC	Reg HEX	0	1	2	3	4	5	6	7	8	9
0000	0x0000	fecha			hora			nº			
0010	0x000A	latitud			longitud			HR			
0020	0x0014	latitud			longitud			HR			
0030	0x001E	latitud			longitud			HR			
...	...	latitud			longitud			HR			
4070	0x0FE6	latitud			longitud			HR			
4080	0x0FF0	latitud			longitud			HR			
4090	0x0FFA	latitud			longitud			HR			

TABLA 3. GESTIÓN EEPROM

Los registros se ordenan en paquetes de 10 bytes, llegando a tener una capacidad de 400. En los cuatro primeros se coloca una variable de tipo 'float', igual que en los cuatro siguientes. Los bytes 8 y 9 de cada trama de 10 corresponden al valor de frecuencia cardiaca, albergado en una variable entera. En la primera fila se escribe la fecha y hora cuando comienza la ruta, los bytes 8 y 9 albergan una variable entera que representa el número de puntos guardados en la última ruta guardada.

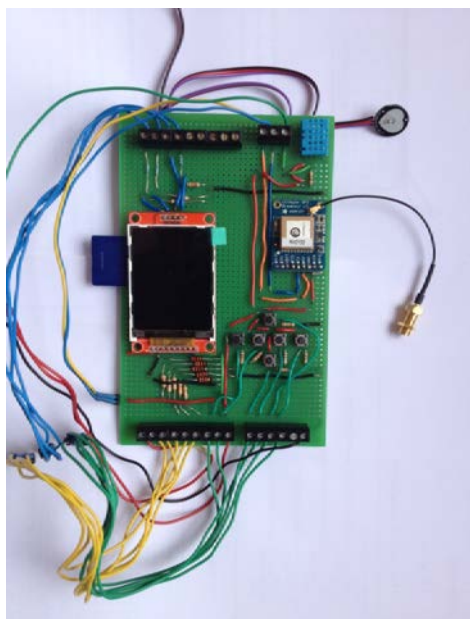


FIGURA 3. IMAGEN DEL PROTOTIPO

2.2 PANTALLA LCD + GESTIÓN

TARJETA SD

El módulo utilizado en este proyecto incorpora una pantalla de 2.2 pulgadas y conectores para una tarjeta SD a través de bus SPI.

El 'display' tiene una resolución 240 x 320 píxeles, con un pin exclusivo para activar la luz interna de la pantalla y poder visualizar los colores. Trabaja con la gama de colores RGB, que se definen como la mezcla de los colores primitivos rojo, verde y azul. Pese a contener millones de colores el tamaño

de la pantalla hace que algunos colores no se diferencian bien. Por eso se han escogido colores

intensos como el rojo, negro y blanco para contrastar bien los datos y poder leer más rápidamente la información.

Para la comunicación serie, se emplean 5 pines (CS, RS, SCL, SDA y RST) que barren cada fila de píxeles, hasta completar la matriz. Todo esto transcurre algo lento para funcionalidades deportivas, pero el compacto diseño y el precio del módulo eran atractivos respecto a otros competidores.

La librería utilizada ha tenido un papel fundamental para el diseño de las pantallas. Ésta, desarrollada por Henning Karlsen hace posible escoger el tamaño de la fuente, el color, la posición o dibujar formas.

El módulo de inserción de la tarjeta de memoria flash acepta el formato SD, de cualquier capacidad. Para la conexión se utilizan cuatro pines estándar (CS, MOSI, MISO y SCK) que son capaces de leer y escribir archivos.

La alimentación de los dos dispositivos del módulo es conjunta. Pese a trabajar con niveles lógicos de 3.3V, el Arduino envía las tramas y a través de un divisor de tensión se reducen los niveles digitales para que puedan ser interpretados por el módulo.

2.3 RECEPTOR GPS

La recepción de señales de posición puede obtenerse mediante diferentes protocolos. El módulo desarrollado por Adafruit se basa en un receptor GPS de la marca Mediatek. Éste envía datos en formato NMEA y espera respuesta por parte del satélite. En cada paquete de información se reciben tramas que incluyen: fecha y hora GMT, latitud y longitud, velocidad instantánea, altura sobre el nivel del mar y detalles sobre la calidad de la señal (satélites activos, algoritmos de corrección, etc.). En la siguiente tabla se describe como ejemplo el contenido de la tramas más utilizada en este proyecto, GPGLA.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14
\$GPGLA	hhmmss.ss	llll.ll	a	yyyyy.yy	a	x	xx	x.x	x.x	M	x.x	M	x.x	xxxx*hh

TABLA 4. INFORMACIÓN OBTENIDA EN GPGLA

1. UTC de la posición
2. Latitud
3. N o S
4. Longitud
5. E o W
6. Indicador de calidad GPS (0=inválido, 1=GPS.fix, 2=Diff.GPS.fix)
7. Número de satélites activos
8. HDOP (Horizontal Dilution of Precision)
9. Altitud de la antena sobre el nivel del mar

10. Unidad de medida de 9 [metros]
11. Separación geoidal
12. Unidad de medida de 11 [metros]
13. ID de la estación de acceso
14. Checksum

Adafruit ofrece una librería que extrae directamente la información de las tramas en variables individuales. Descompone la información y elimina los bits de control. Además, ofrece la posibilidad de mejorar la potencia de recepción con una antena externa. La antena que se ha incorporado en el proyecto, está compuesta de un cable coaxial y una etapa de ganancia. Permite también colocar el punto de recepción en algún lugar más visible como la parte trasera de la bicicleta.

2.4 SENSOR DE FRECUENCIA CARDIACA

Uno de los sectores de investigación más vivos es el de la electrónica para aplicaciones médicas. Existen multitud de proyectos que incorporan periféricos para recoger información corporal como alcoholímetros, electrocardiogramas, tensión sanguínea o sensores que miden la temperatura de la sangre. Proyectos con carácter más industrial y otros más académicos, como este.

Los entrenamientos deportivos a día de hoy se preparan con mucho detalle, estableciendo planes de complejas estrategias para poder mejorar el rendimiento deportivo. Por eso, en este proyecto se ha incluido este sensor, ya que da la oportunidad de marcar ritmos en función de un valor objetivo como lo es el pulso cardiaco, añade la personalización de la ruta para sacar el máximo provecho del entrenamiento.

La primera idea para incorporar un sensor de este tipo fue adaptarse a un protocolo de comunicaciones ya establecido para poder soportar varios dispositivos. Se hicieron investigaciones en las bandas de frecuencia cardiaca más conocidas y se distinguieron dos tipos de protocolos, 'bluetooth' y ANT+.

ANT+ ha conseguido, en pocos años, colocarse como el sistema de comunicaciones puntero en lo que a deportivo se refiere. Se ofrecen al mercado más de 400 sensores que interoperan entre sí, cada uno para su función.

Garmin, dueña del protocolo ANT+, facilita la comunicación de dispositivos con documentación y simuladores que ofrece en su página web. Tras varias pruebas, se estableció la comunicación con los sensores de las marcas Garmin y Suunto desde el ordenador. A través de una antena USB y la interfaz ANTware II (que se muestra en la Figura 4) se enviaron las tramas necesarias para establecer los parámetros que crean la red y permiten comunicar a los dispositivos.

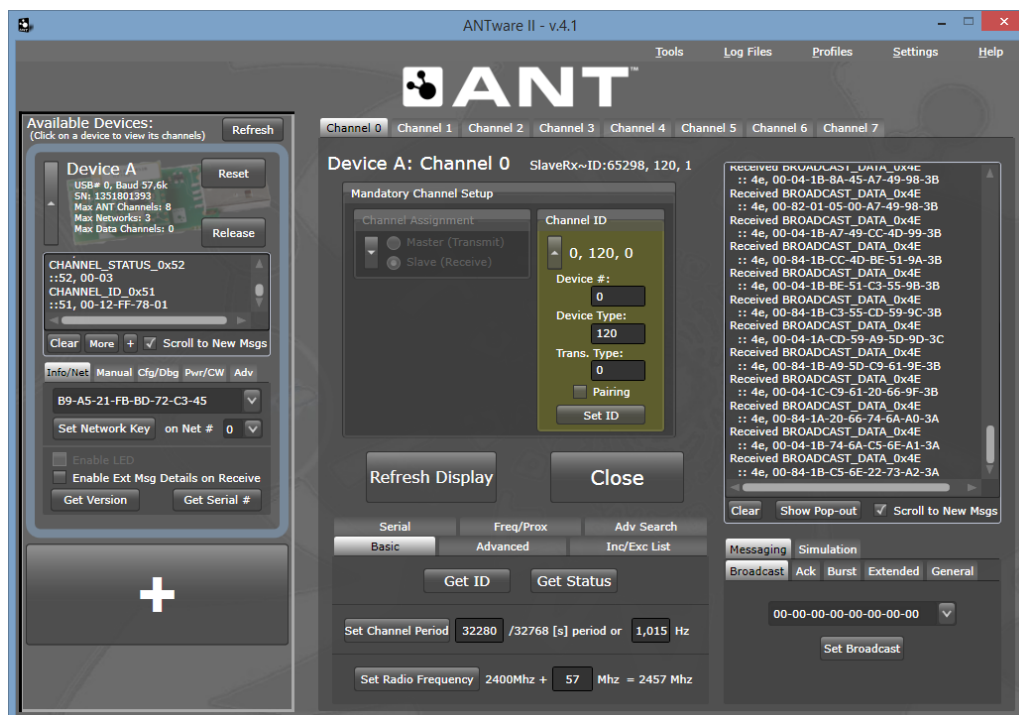


FIGURA 4. INTERFAZ DE COMUNICACIÓN ANT

En cambio, con un módulo desarrollado por Sparkfun, que incorpora el circuito de comunicación ANT+ de Nordic Semiconductors (NRF24AP1), se enviaron las mismas tramas desde el Arduino, pero nunca se recibió respuesta de los sensores. Tras contactar con el personal de ANT+ no se consiguió entablar la comunicación. Por esto se decidió incorporar la medida con otro tipo de sensor, el oxímetro.

El color de la sangre varía dependiendo de lo saturada de oxígeno que se encuentre. Cuando la molécula de hemoglobina libera oxígeno deja pasar menor luz infrarroja por adquirir un leve tono azul. Mediante la reflexión de luz infrarroja en la piel a lo largo del tiempo, se podrán observar los valores máximos de luz reflejada. Estos valores corresponderán con los ciclos de bombeo de la sangre. Analizando en el tiempo, esos valores máximos de una señal analógica como la de la Figura 5, se podrá obtener el número de pulsaciones a las que opera el corazón.

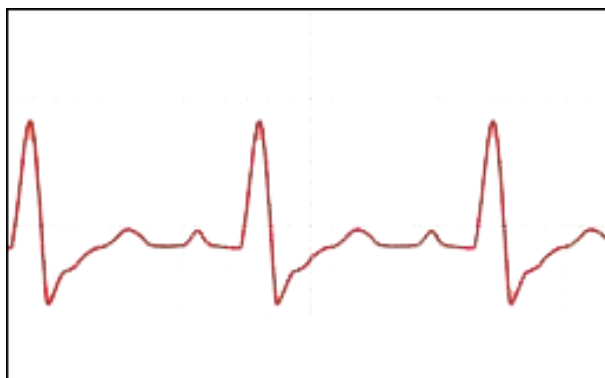


FIGURA 5. SEÑAL PULSO CARDIACO

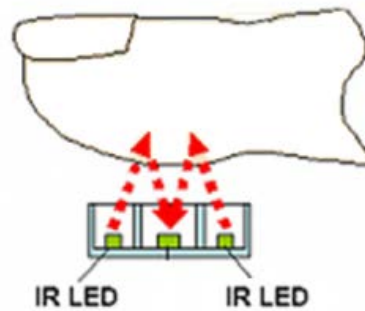


FIGURA 6. ESQUEMA SENSOR PULSACIONES/MIN

El Arduino recoge muestras cada 2ms a través de una interrupción del timer2. Acumulando 250 muestras y seleccionando los extremos, realiza la siguiente operación:

$$\text{nº maximos} / \text{tiempo} = \text{pulsaciones}$$

Escalando el resultado a segundos se obtiene el número de pulsaciones por minuto, o en inglés BPM.

2.5 SENSOR DE TEMPERATURA

El simple uso de un termistor en un circuito analógico hubiese funcionado para las medidas de temperatura. Sin embargo, la precisión y formato del sensor DHT 11 fueron determinantes en esta elección, dando además la oportunidad de exponer también la humedad relativa. Pese a que en la primera versión no se utilice la humedad, con pocas líneas de código se puede mostrar, dejando al gusto del usuario esta característica.

La obtención de datos se obtiene a través de comunicación serie. El fabricante del sensor pone a disposición una librería que permite comprobar el estado del sensor y obtener las medidas a través de funciones ya implementadas.

Con esta medida se quiere dar algo de importancia al entorno donde se desarrolla la ruta. En la llegada a una cumbre por ejemplo, la temperatura puede ser un dato interesante para anotar cómo responde el cuerpo en estas condiciones.

2.6 MANDOS - BOTONES

La interfaz de la máquina con el usuario debe ser fácil. Por esta razón desde el primer momento se rechazaron las pantallas táctiles y se optó por una botonadura amplia. El uso de guantes y el pequeño tamaño de la pantalla no permitía una interfaz táctil decente.

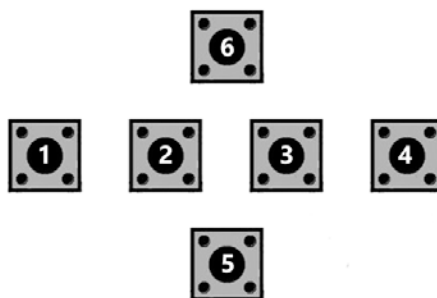


FIGURA 7. PULSADORES PROTOTIPO

El número de pulsadores utilizados son seis en la disposición que muestra la Figura 7. Pese a haber podido reducir el número de botones incluyendo duplicidad de funciones en cada mando, se quiere dejar que el usuario pueda tranquilamente incluir las suyas propias. Una de las tareas no llevadas a cabo es la opción de ampliar el dibujo de la ruta.

Eléctricamente se han instalado como 'pull-down', es decir, por defecto a nivel bajo (Figura 8). Añadiendo una resistencia de 10k conectada a masa, permite fijar los valores lógicos de la señal entrante al microcontrolador.

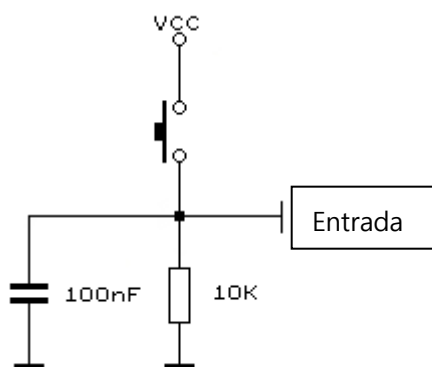


FIGURA 8. PULL DOWN PULSADORES

2.7 SOPORTE

Se ha diseñado un soporte para dar mayor robustez al equipo, sobre todo durante las pruebas. Se compone de dos partes, tapa inferior y superior. Es un diseño sencillo pues pretende ser temporal, como muestra el plano de explosionado de la Figura 9.

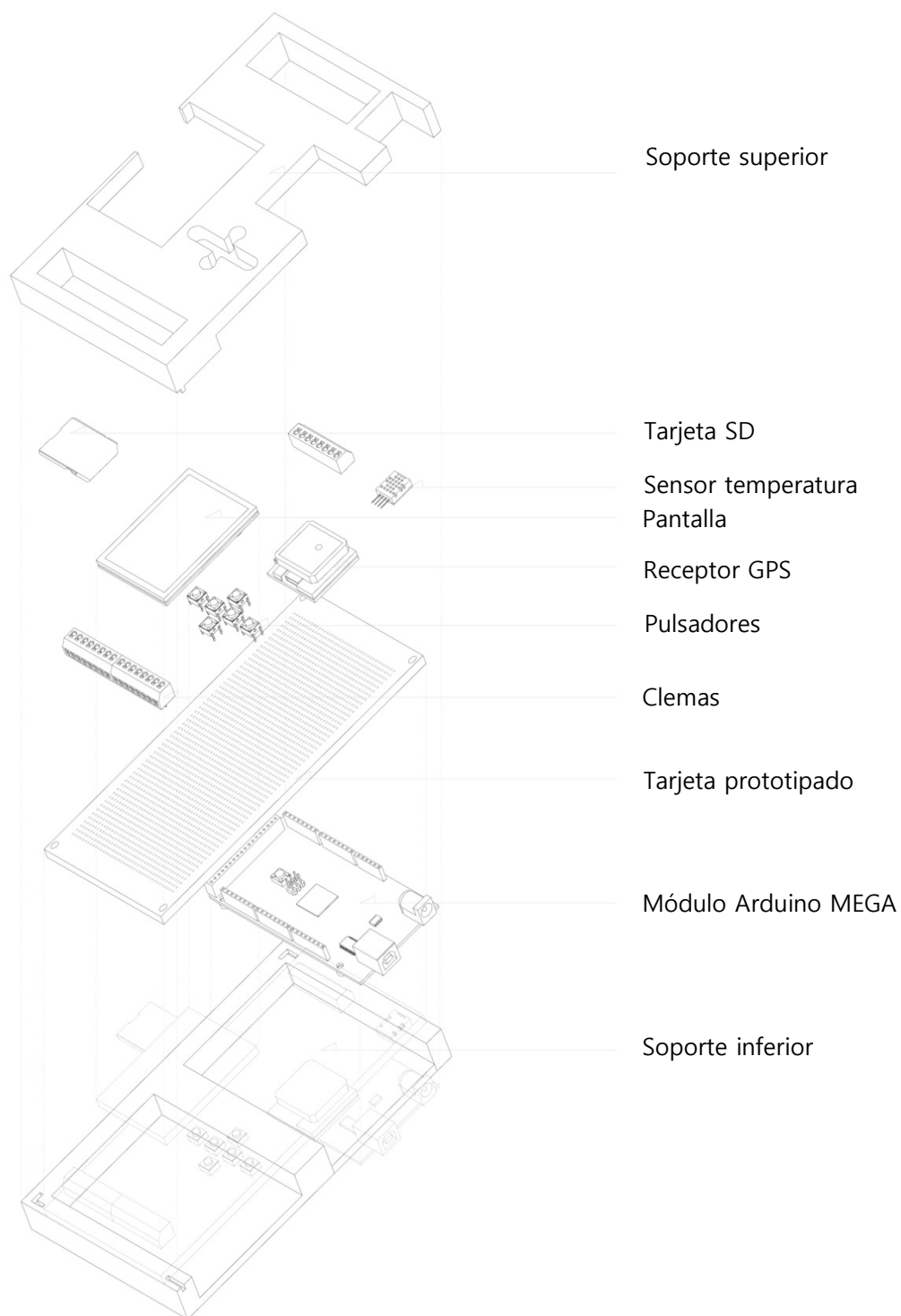


FIGURA 9. DETALLE EXPLOSIONADO PROTOTIPO

Su fabricación se ha llevado a cabo con la tecnología de fabricación FDM, Modelado por Deposición Fundida. La impresión 3D, como es conocida esta técnica, obtiene buena definición (100um) en un corto periodo de tiempo (tiempo de impresión de ambas partes, 12h).

3. SATBIKE

Se denomina SATBIKE al prototipo de ordenador de a bordo realizado. Se encuentra montado en la bicicleta y desde ahí, interactúa con el deportista. Aquí, se describirá el funcionamiento general del sistema, las variables que permiten realizar las transiciones en la pantalla, mientras se recogen las diferentes medidas de los sensores.

3.1 DIAGRAMA DE ESTADOS

El sistema está preparado para trabajar en dos versiones, siguiendo una ruta anteriormente preestablecida o en modo 'libre', es decir, sin ruta a seguir, simplemente recogiendo valores instantáneos del trayecto realizado.

Tras encender el dispositivo se expone un mensaje que presenta si la recepción de la ruta desde la tarjeta SD, ha sido efectuada. Tras esta distinción, se dirige al usuario a la pantalla correspondiente. A continuación se presenta el diagrama de estados en la imagen 10 y se explicará con más detalle la operatividad de los dos modos.

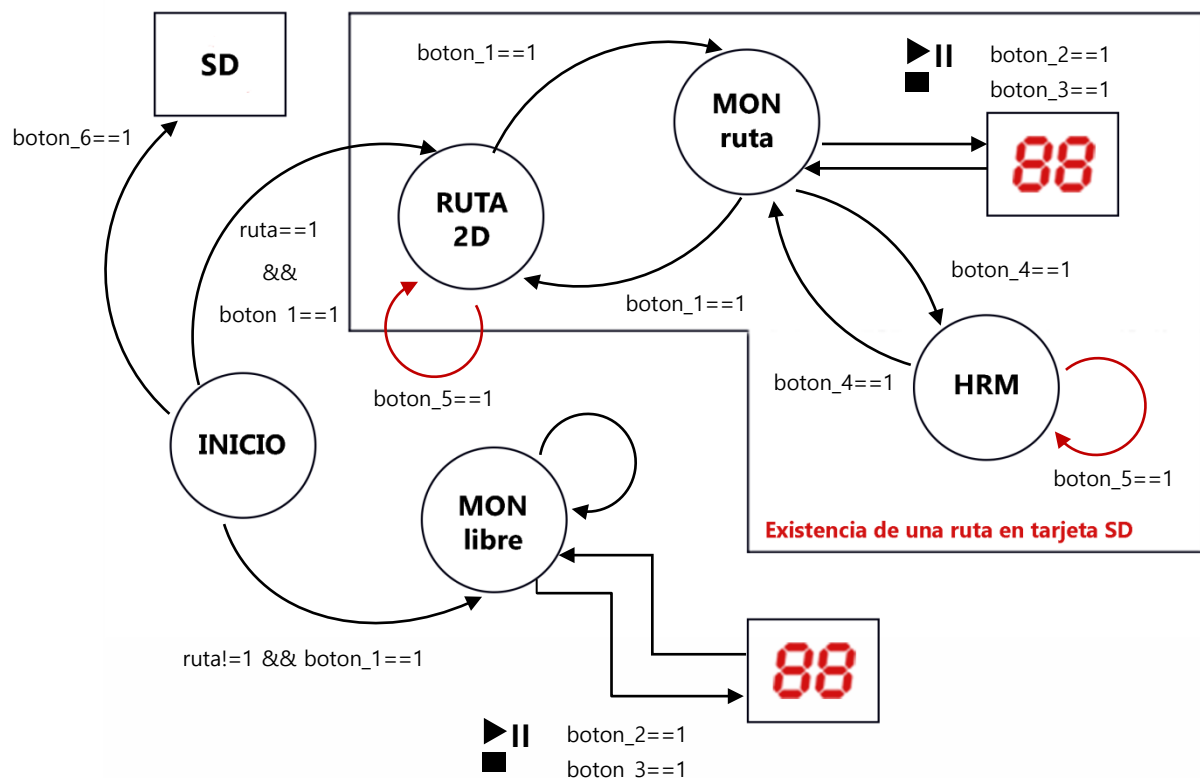


FIGURA 10. DIAGRAMA DE ESTADOS

3.1.1 MODO LIBRE

Al modo libre se accede directamente cuando no se ha introducido una ruta en la tarjeta de memoria. En esta pantalla, el usuario podrá comenzar un trayecto e ir almacenando los valores en la EEPROM. Más tarde, con una tarjeta de memoria, podrá extraerlos.

En la única pantalla visible se muestra la siguiente información:

- Altitud [m]
- Distancia recorrida [m]
- Velocidad instantánea [km/h]
- Velocidad media [km/h]
- Hora
- Tiempo [s]
- N° de pulsaciones por minuto fijadas
- Temperatura [°C]
- Identificador de la calidad del GPS



FIGURA 11. PANTALLA MODO LIBRE

Una vez comenzado el cronómetro los valores empiezan a aparecer en pantalla dispuesta como muestra la Figura 11. Cada 30s se muestrea para guardar en la memoria no volátil, las coordenadas y la pulsación asociada al punto de paso.

3.1.1 MODO RUTA

Introducido un itinerario en la tarjeta de memoria aparece en pantalla el mensaje 'Ruta encontrada'. Desde aquí el dispositivo comienza a trabajar con funcionalidad plena.

El código está preparado para trabajar con rutas que contengan coordenadas en toda Europa y parte de África y Asia, concretamente en el sector que se muestra en la Figura 12. La distancia de la ruta tampoco es problema, pues el algoritmo de escala se adecúa al tamaño de la pantalla mostrando la totalidad del recorrido, obviamente con menos detalle.

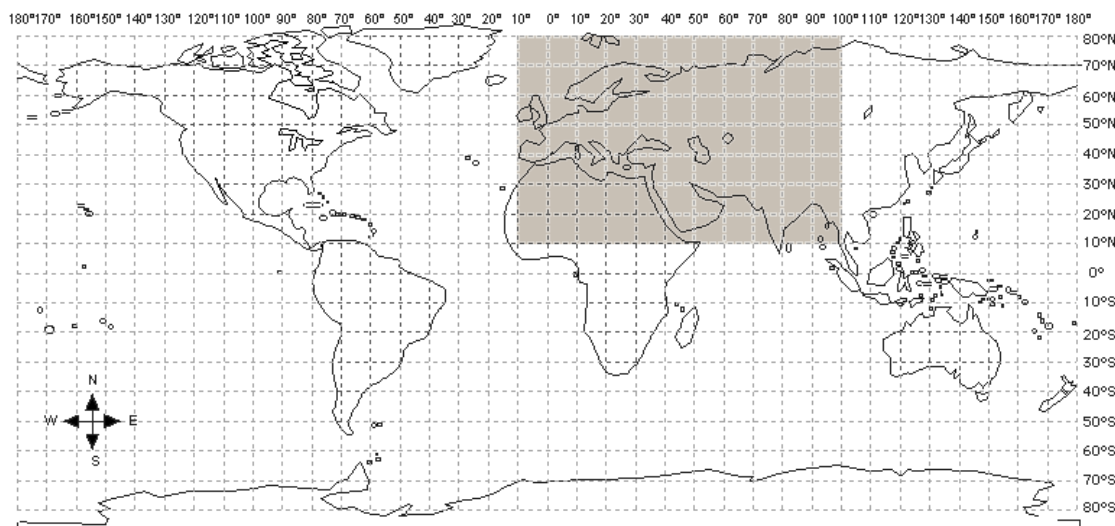


FIGURA 12. MAPA FUNCIONALIDAD RUTAS

El usuario tendrá la opción de visualizar 4 pantallas:

a) Pantalla inicio



FIGURA 13. INICIO

Da la bienvenida al usuario, le informa de la calidad de la señal de posicionamiento, muestra la fecha y hora actual, expone la temperatura y diferencia entre modo libre o ruta. Además desde este estado, se podrán transferir los datos a un archivo en la tarjeta de memoria que tendrá el nombre de la fecha en la que se realizó el camino.

b) Pantalla ruta

En color blanco dibuja mediante rectas la propuesta de ruta fijada. Además, señala con un cuadrado la posición en tiempo real del dispositivo. Una vez que la ruta se ha comenzado a desarrollar (iniciando el cronómetro) se podrá pulsar el botón 5, que mostrará en azul el recorrido realizado.

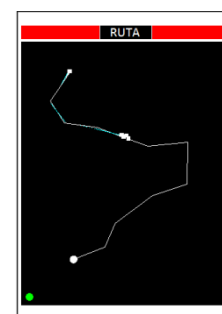


FIGURA 14. RUTA

c) Pantalla monitorización

Los valores numéricos se exponen aquí. El cronómetro se gestiona exclusivamente desde esta pantalla y permite pausar y detener aquellas medidas de tiempo. La información que se muestra con sus unidades es:

- Altitud [m]
- Velocidad instantánea [km/h]
- Distancia recorrida [m]
- Distancia a meta [m]

- Hora
- Tiempo
- N° pulsaciones por minuto fijadas
- N° pulsaciones por minuto reales
- Temperatura [°C]
- Identificador de la calidad del GPS



d) Pantalla HRM

FIGURA 15. MONITORIZACIÓN

A esta pantalla se accede pulsando el botón 4 en el estado monitorización. Añade en un gráfico la evolución de frecuencia cardíaca prefijada a lo largo de la ruta, mientras también se muestra el perfil de altitudes. Se incorporan ejes a 50, 100 y 150 pulsaciones por minuto. Como en la pantalla ruta, el usuario con el pulsador 5 solapa las pulsaciones realizadas con las previamente asignadas.

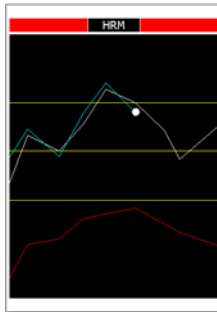


FIGURA 16. HRM

4. SAT_RUTAS

Las rutas que lee el ciclocomputador deben estar en un formato especial. Este formato se genera desde un programa, desarrollado también para este proyecto, denominado SAT_Rutas. La razón de este cambio de formato se realiza para agilizar la lectura por parte del microcontrolador y así poder acceder al archivo más rápidamente.

Con la misma interfaz se ha aprovechado para personalizar una ruta incluyendo un entrenamiento basado en la frecuencia cardiaca.



FIGURA 17. TRANSCRIPCIÓN SAT_RUTAS

4.1 FORMATO GPX

El formato GPX es común en aplicaciones de rutas basadas en localización. Consiste en la descripción de los puntos (waypoints) que conforman la ruta mediante coordenadas (latitud, longitud y altura). Existen múltiples programas y sitios web que permiten crear este tipo de archivos: la más utilizada para las pruebas ha sido Sport Route Planner.

Se añade aquí un ejemplo de ruta en formato GPX para mostrar como los valores numéricos se entremezclan con texto que dificulta la selección de información.

```

<?xml version="1.0" encoding="UTF-8"?>
<gpx version="1.0" creator="SATB - ICAI 2014"
xmlns:xsi="Based on Sport Route Planner"
xmlns="http://www.topografix.com/GPX/1/0"
xsi:schemaLocation="Ruta Ejemplo extension GPX">
  <trk>
    <number>1</number>
    <trkseg>
      <trkpt lat="36.27468445279199" lon="-5.28441309928894">
        <ele>3</ele>
      </trkpt>
      <trkpt lat="36.27397519237707" lon="-5.284906625747681">
        <ele>6</ele>
      </trkpt>
      <trkpt lat="36.27349081570802" lon="-5.285528898239136">
        <ele>3</ele>
      </trkpt>
      <trkpt lat="36.274874741075365" lon="-
5.290356874465942">
    ...
  
```

TABLA 5. CONTENIDO FORMATO GPX

4.1 ARCHIVO SALIDA

El formato de texto que se obtiene tiene la siguiente sintaxis:

```
maxlat: 36,2898918619705
minlat: 36,273490815708
maxlon: -5,28441309928894
minlon: -5,31398177146912
Tdist: 05383
WAYpoint000
36,27468445
-5,28441310
0003
080
WAYpoint001
36,27397519
-5,28490663
0006
082
WAYpoint002
36,27349082
-5,28552890
0003
090
...
```

TABLA 6. SATB.TXT

Las primeras cinco líneas aportan información general sobre el conjunto de la ruta. Valor máximo de latitud, longitud y el valor mínimo. Esto permite ajustar la pantalla sin tener que hacer más operaciones internas. La línea Tdist añade la distancia en metros del total de la ruta. A partir de ahí se muestra la información de cada punto de paso en el siguiente orden: latitud, longitud, altura y frecuencia cardiaca asociada. En la Figura 18 se presenta la interfaz de la aplicación

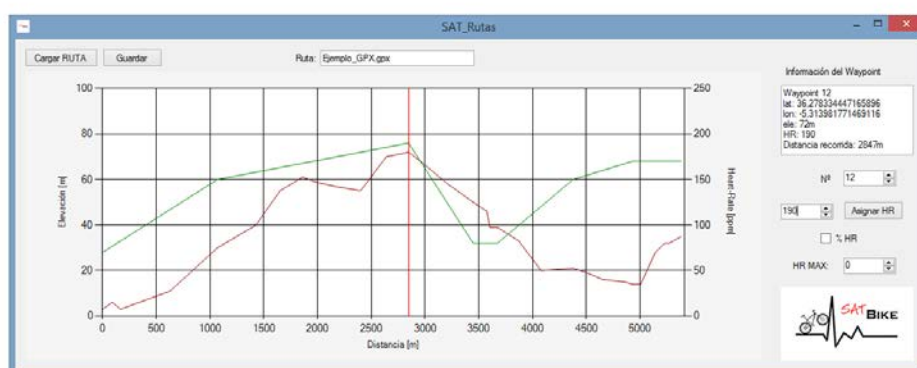


FIGURA 18. INTERFAZ SAT_RUTAS

5. RESULTADOS

Tras las pruebas en campo, con rutas de distintos tipos se han obtenido resultados satisfactorios. La operabilidad de este dispositivo es suficientemente potente para deportistas centrados en el ciclismo. Su bajo consumo hace posible trabajar más de 8 horas a pleno rendimiento y su precisión permite extraer valores para su posterior análisis (error de posición inferior a los 10m).

Se ha recorrido una ruta de 1,2km y en la Figura 19 se comparan la ruta prevista y la realizada. En la parte de la izquierda se presenta el diseño de la ruta creada, formato GPX. En la derecha aparecen los marcadores que han sido obtenidos recogiendo la EEPROM en la tarjeta de memoria.

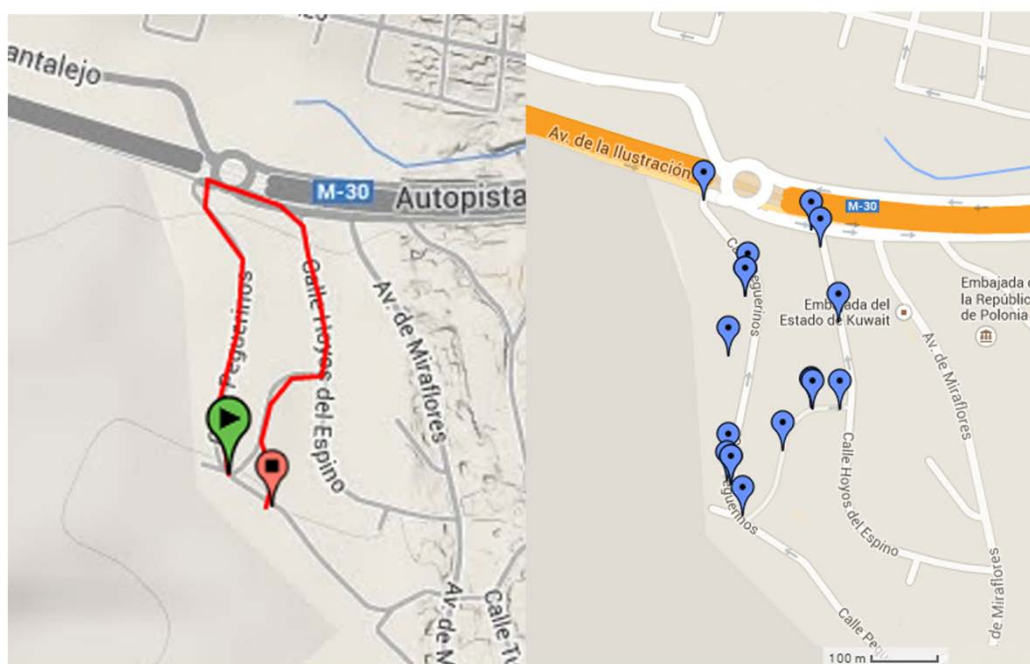


FIGURA 19. COMPARACIÓN RUTAS PRUEBA

El prototipo de ordenador de a bordo es capaz de operar en dos modos: con o sin ruta precargada. Interactuar con información escogida por el usuario y fácilmente insertable con una tarjeta de memoria, capacitan a cualquier persona maniobrar con este producto. A través de seis sencillos pulsadores, el usuario gestiona la información que desea mostrar sobre la ruta.

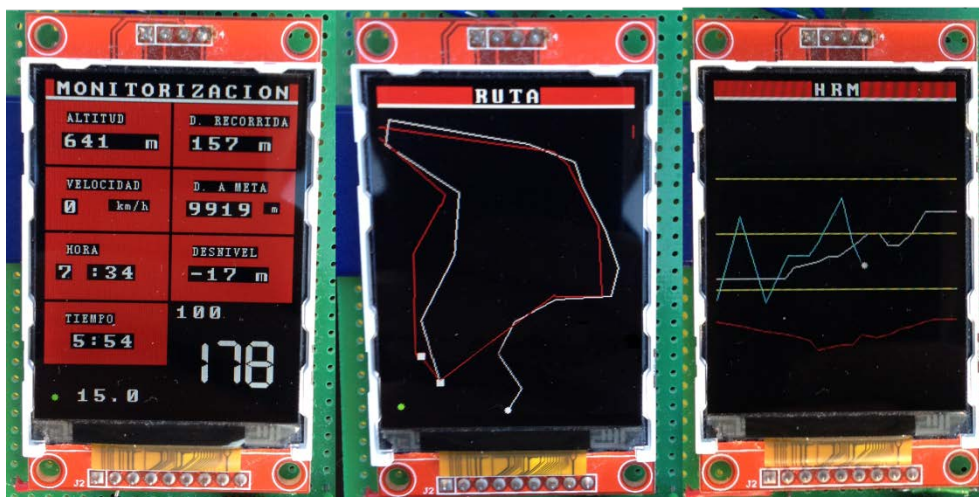


FIGURA 20. PANTALLAS PRINCIPALES

El usuario podrá escoger entre tres pantallas, como muestran las de la ruta probada en la Figura 20. En la pantalla de monitorización se presentan valores numéricos sobre el estado del entrenamiento, Como son la distancia recorrida o el desnivel acumulado. Desde aquí se puede acceder a la tercera pantalla que expone el perfil de altitudes y la variación en las pulsaciones del corredor. En la segunda, se muestra la ruta a la que se superpone el tramo ya realizado.

Uno de los resultados obtenidos ha sido el programa que gestiona una ruta estándar en formato GPX, capaz de interpretar la sintaxis de este archivo y conseguir extraer la información contenida. El valor diferenciador, lo añade la posibilidad de poder personalizar las rutas con un plan de entrenamiento cardíaco. Con la misma interfaz, se consigue cumplir ambos objetivos.

El diseño de sujeción actual es lo suficientemente rígido para aguantar los movimientos de una bicicleta de montaña, pero al reducir el tamaño mejorará en este aspecto.

6. CONCLUSIONES

Pese a la infinidad de aplicaciones de móvil que ofrecen los servicios aquí descritos, los deportistas de hoy quieren aparatos exclusivos, más personalizables, de menor consumo, más ergonómicos y con mayor precisión. En este proyecto se ha realizado un prototipo que con la personalización de rutas añade valor a las ciclocomputadoras corrientes. El tamaño es grande como se muestra en la Figura 21, no obstante, con las funcionalidades conseguidas reducir las dimensiones de los componentes es tarea pendiente para el segundo prototipo.

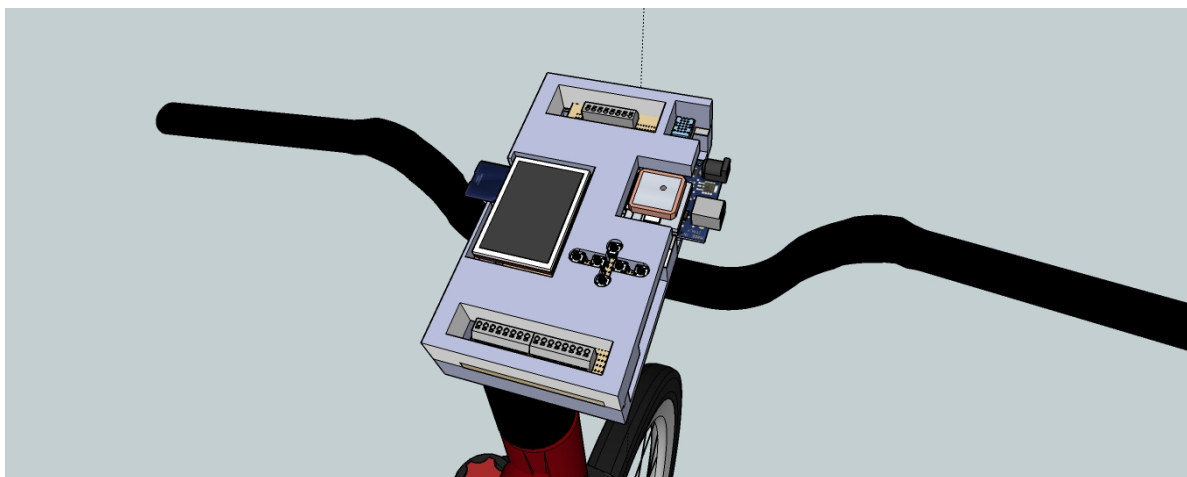


FIGURA 21. SIMULACIÓN 3D DEL PROTOTIPO

Por ejemplo, el sensor de frecuencia cardiaca resulta algo incómodo. Su uso enganchado al dedo, con los movimientos de una bicicleta de montaña hace que la medida sea imprecisa. Una alternativa, es incorporar el sensor a la parte frontal de un casco, ya existe algún sensor colocado aquí con resultados positivos. El hecho de que la pantalla tarde demasiado tiempo en realizar las transiciones, o realizar un diseño más compacto al actual (Figura 21), con la fabricación de una tarjeta con el circuito impreso, son algunos otros ejemplos.

Las empresas tecnológicas desarrollan constantemente nuevas funcionalidades, por los diferentes gustos del consumidor o por la diferenciación de cada deporte. Crear un prototipo escalable y moldeable para cada deportista parece interesante y que seguro tendría cabida en un mercado tan competido. Con este dispositivo se han sentado las bases para poder personalizar los entrenamientos y recortar segundos.

7. FUTUROS DESARROLLOS

Sentadas las bases descritas en la totalidad del proyecto nacen nuevas ideas para perfeccionar el dispositivo.

7. 1 MODO INDIVIDUAL

En cuanto a hardware se refiere, se pueden implementar mejoras en los siguientes aspectos:

a. Diseño de una tarjeta específica que optimice el esquema de conexionado. Así, la colocación en el manillar de la bicicleta será más sencilla, mejorará el peso del dispositivo y hará más ergonómico su diseño.

b. La pantalla que se utiliza actualmente es demasiado lenta para una ruta en bicicleta donde la posición o velocidad varían cada segundo. Se debería incorporar otra interfaz SPI para poder gestionar el 'display' y la tarjeta de memoria.

c. El sensor de pulso cardíaco debería ser inalámbrico o estar incorporado en algún punto de la bicicleta que facilite su uso. En el manillar como en bicicletas estáticas o en la parte frontal del casco. Aunque la mejor opción es poder afiliarse a protocolos más conocidos y aumentar el número de sensores a utilizar.

d. Incorporación de sensores nuevos, como un giróscopo o un sensor de cadencia. Aumentar las variables que definen una ruta y poder completar el entrenamiento. Un acelerómetro podría obtener con más precisión distancias y velocidades instantáneas.

e. Incorporar señales de prevención de riesgos. Con un giróscopo poder enviar un mensaje de ayuda con la caída del ciclista.

d. Posibilitar al algoritmo de ruta aumentar la zona geográfica. Conseguir hacer zoom y moverse en el dibujo de un itinerario.

Pese a que las funcionalidades están dirigidas hacia el mundo de las dos ruedas. La misma estrategia se podría aplicar a otras destrezas como montañismo, 'footing' o deportes en el mar como vela, kitesurf o windsurf.

7. 2 MODO GRUPO

Con una red de dispositivos del mismo tipo, se podría intercambiar información en tiempo real, creando equipos de entrenamiento donde unos y otros mejoren sus marcas tirando del grupo. Con el envío de señales de radio se planearían rutas compartidas.



Otra aplicación de comunicación podría ser enviar los datos a una central. Albergada una base de datos recibe el detalle de todos los ciclistas. Un médico que controle a todos los ciclistas de su equipo a la vez.

Por ejemplo un desarrollo de este tipo se ha llevado a cabo en el proyecto titulado "Desarrollo de sistema de comunicaciones para una bicicleta de montaña" que interconecta radios albergadas en bicicletas.

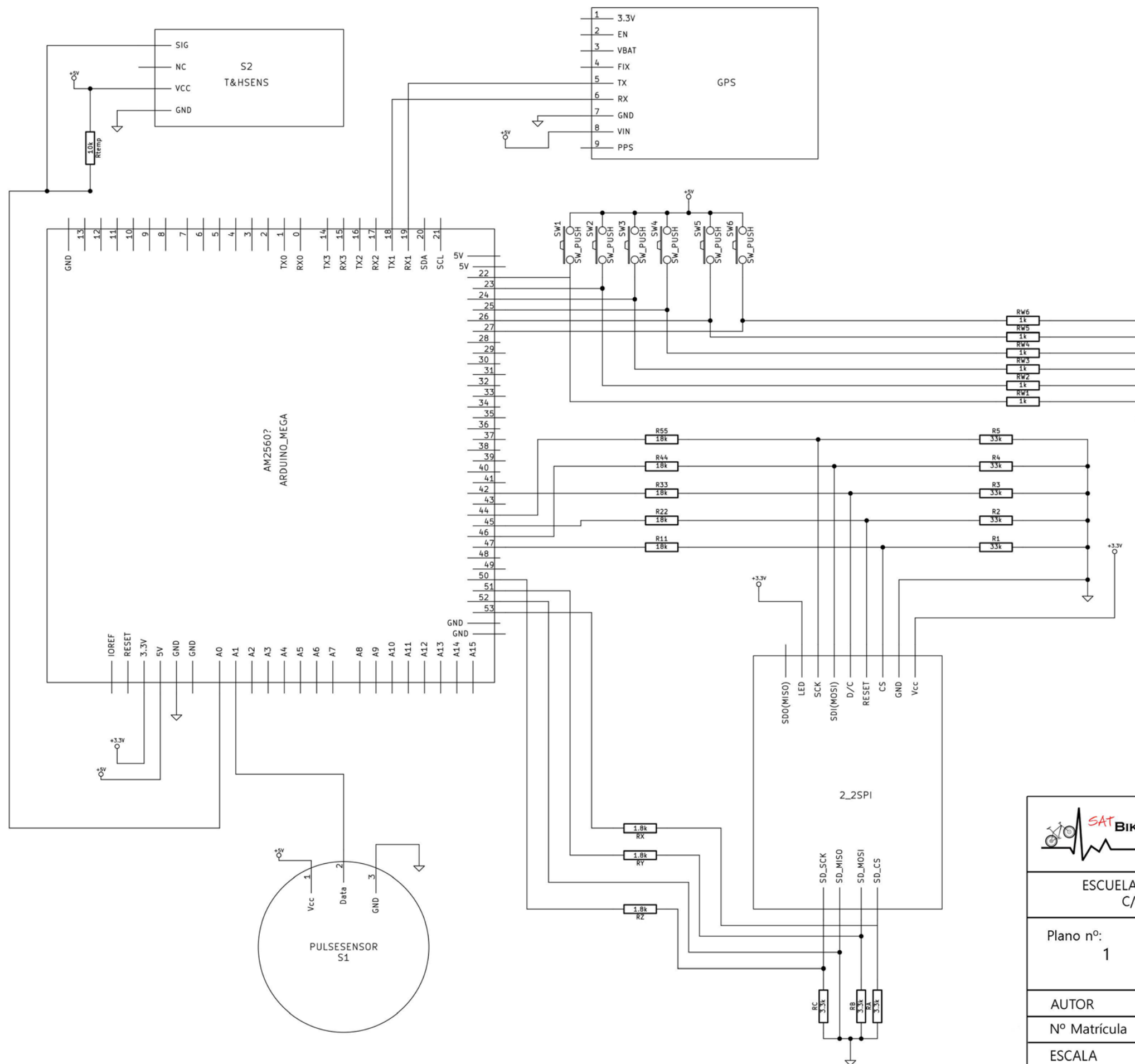


8. REFERENCIAS

- [1] N.S.A.Zulkifli, F.K.Che Harun, and N.S.Azahar - "Xbee Wireless Sensor Networks for Heart Rate Monitoring in Sport Training"
- [2] M. Betancur, A. Jaramillo, M. Ospina, B. Moreno, A. Tejada - "Diseñar e Innovar el pulsioxímetro de forma creativa y práctica para obtener resultados efectivos a la hora de la lectura de sus datos como la oxigenación, oxihemoglobina, metahemoglobina" Sep. 2011
- [3] L. Aguirre Valencia - "Diseño y construcción de un oxímetro de pulso mediante la utilización de una plataforma de automatización momentum" 2007
- [4] A. Yarlalagadda (Digi-Key) - "Designing a Wireless Heart Rate Monitor with Remote Data Logging" Feb. 2012
- [5] C. Cupeiro Durán, J. López González, D. Redondo Arroyo, "Ordenador de a bordo (visualización de velocidad, distancia, temperatura...)", Universidad Rey Juan Carlos, 2012
- [6] T. Basagoiti Fernández de la Puente - "Automatización de una casa a escala. Gestión de energía", May. 2013
- [7] Ron/Spinning magnets - Electricbike "Qoros eBIQE, Exotic Concept E-Bike", Mar. 2014
- [8] Z. Romano, "DIY bicycle computer with Arduino", Ago. 2013
- [9] E. Terol, - Ecoavant "La bicicleta conquista Europa", Oct. 2013
- [10] J. Leber - "A Shrinking Garmin Navigates the Smartphone Storm" Mar.2013
- [11] ANT Message Protocol and Usage, Ene. 2013
- [12] NMEA - [http:// aprs.gids.nl/nmea/](http://aprs.gids.nl/nmea/)
- [13] Wikiloc - <http://es.wikiloc.com/wikiloc/home.do>
- [14] Manual Garmin Edge 510 - <http://www.garmin.com/es-ES>
- [15] A Runner's Wearable Display - <http://checkmybeat.wordpress.com/>
- [16] GPX - The GPS Exchange Format - <http://www.topografix.com/gpx.asp>

II. PLANOS

A continuación se adjuntará el esquema de conexión. No se incluirán los planos de los sensores y otros periféricos por estar incluidos en la hoja de características.



 		DESARROLLO DE UN PROTOTIPO DE ORDENADOR PARA UNA BICICLETA DE MONTAÑA	
ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA - ICAI C/ Alberto Aguilera 23 - 28015 Madrid			
Plano nº: 1		ESQUEMA CONEXIONADO	
AUTOR		Íñigo Fernández de Aroz García-Miñaur	
Nº Matricula		201001416	
ESCALA		Sin escala (plano de conexionado)	
Fecha: 08/07/2014		Firma: 	

III. PRESUPUESTO

1. MEDICIONES

En este apartado se recogen las unidades de los componentes y equipos utilizados, así como las horas de trabajo imputables al proyecto.

1.1 COMPONENTES PRINCIPALES

Componentes	Cantidad
Arduino MEGA 2560	1
Placa Board	1
Pantalla TFT SPI 2.2"	1
Módulo GPS MTK3339	1
Antena GPS con base magnética	1
Sensor de ritmo cardiaco	1
Sensor de temperatura y humedad DHT 11	1
Pulsadores	6
Resistencias	23
Cable (m)	4

TABLA 7. RELACION DE COMPONENTES

1.2 EQUIPO Y HERRAMIENTAS

Componentes	Cantidad	Horas de proyecto	Horas de uso al año
Ordenador	1	500	1600
Kit Soldadura	1	2	40
Polímetro	1	1	450
Batería 5200mAh	1	10	200

TABLA 8. RELACIÓN DE EQUIPOS Y HERRAMIENTAS

1.3 SOFTWARE

Programa	Cantidad	Horas de proyecto	Horas de uso al año
Arduino IDE	1	750	1000
Microsoft Visual Studio 2009	1	150	800
Microsoft Office 2007	1	200	2000
KiCad	1	50	60
Sketchup Pro	1	50	200

TABLA 9. RELACIÓN DE HERRAMIENTAS SW

1.4 MANO DE OBRA DIRECTA

Actividad	Horas
Programación	400
Pruebas y solución de problemas	120
Documentación del proyecto	100

TABLA 10. RELACIÓN DE HORAS DE MANO DE OBRA

2. PRECIOS UNITARIOS

En este apartado se indican los precios de cada uno de los elementos necesarios para llevar a cabo este proyecto, los cuales se han citado anteriormente.

2.1 COMPONENTES PRINCIPALES

Componentes	Precio (€/u)
Arduino MEGA 2560	41,00
Placa Board	4,49
Pantalla TFT SPI 2.2"	13,36
Módulo GPS MTK3339	31,10
Antena GPS con base magnética	10,80
Sensor de ritmo cardiaco	18,50
Sensor de temperatura y humedad DHT 11	4,50
Pulsadores	0,20
Resistencias	0,10
Cable (m)	0,60

TABLA 11. PRECIO UNITARIO COMPONENTES PRINCIPALES

2.2 EQUIPO Y HERRAMIENTAS

Componentes	Precio (€/u)
Ordenador	620,00
Kit de Soldadura	11,50
Polímetro	39,90
Batería 5200mAh	29,99

TABLA 12. PRECIO UNITARIO EQUIPOS Y HERRAMIENTAS

2.3 SOFTWARE

Programa	Precio (€/u)
Arduino IDE	Libre
Microsoft Visual Studio 2009	800,00
Microsoft Office 2007	101,18
KiCad	Libre
Sketchup Pro	378,00

TABLA 13. PRECIO LICENCIAS SW

2.4 MANO DE OBRA DIRECTA

Actividad	Precio (€/u)
Programación	25,00
Pruebas y solución de problemas	60,00
Documentación del proyecto	42,00

TABLA 14. PRECIO UNITARIO MANO DE OBRA

3. SUMAS PARCIALES

En este capítulo se muestran los importes parciales de cada uno de los elementos que componen las distintas categorías.

3.1 COMPONENTES PRINCIPALES

Componentes	Cantidad	Precio (€/u)	Coste total (€)
Arduino MEGA 2560	1	41,00	41,00
Placa Board	1	4,49	4,49
Pantalla TFT SPI 2.2"	1	13,36	13,36
Módulo GPS MTK3339	1	31,10	31,10
Antena GPS con base magnética	1	10,80	10,80
Sensor de ritmo cardiaco	1	18,50	18,50
Sensor de temperatura y humedad DHT 11	1	4,50	4,50
Pulsadores	6	0,20	1,20
Resistencias	23	0,10	2,30
Cable (m)	4	0,60	2,40
TOTAL			129,65

TABLA 15. SUMAS PARCIALES COMPONENTES PRINCIPALES

3.2 EQUIPO Y HERRAMIENTAS

Componentes	Unidades	Horas de proyecto	Horas por año	Precio (€/u)	Amortización anual	Coste (€)
Ordenador	1	500	1600	620,00	25%	48,44
Kit Soldadura	1	2	40	11,50	25%	0,14
Polímetro	1	1	450	39,90	25%	0,02
Batería 5200mAh	1	10	200	29,99	25%	0,38

TOTAL	49,00
-------	-------

TABLA 16. SUMAS PARCIALES EQUIPOS Y HERRAMIENTAS

3.3 SOFTWARE

Programa	Uds.	Horas de proyecto	Horas por año	Precio (€/u)	Amortización anual	Coste (€)
Arduino IDE	1	750	1000	0	-	0
Microsoft Visual Studio 2009	1	150	800	800	20%	30,00
Microsoft Office 2007	1	200	2000	101,18	20%	2,02
KiCad	1	50	60	0	-	0
Sketchup Pro	1	50	200	378,00	20%	18,90
TOTAL						50,92

TABLA 17. SUMAS PARCIALES SW

3.4 MANO DE OBRA DIRECTA

Actividad	Horas	Precio (€/hora)	Coste total (€)
Programación	400	25,00	10.000,00
Pruebas y solución de problemas	120	60,00	7.200,00
Documentación del proyecto	100	42,00	4.200,00
TOTAL			21.400,00

TABLA 18. SUMAS PARCIALES MANO DE OBRA DIRECTA

4. PRESUPUESTO GENERAL

Sumando la contribución de todas las categorías anteriores se concluye que el coste del proyecto, impuestos incluidos, asciende a:

Concepto	Coste (€)
Componentes principales	129,65
Equipo y herramientas	49,00
Software	50,92
Mano de obra directa	21.400,00
TOTAL	21.629,57

TABLA 19. PRESUPUESTO GENERAL

IV. MANUAL DE USUARIO

1. GESTIÓN DE RUTA

El programa desarrollado para este proyecto tiene como dos objetivos principales:

- a. Introducción de metas para el entrenamiento en valores de frecuencia cardíaca.
- b. Obtener los datos de una ruta para optimizar la lectura del archivo por parte del microcontrolador.

1.1 INSTALACIÓN

Para poder ejecutar este programa, a partir de ahora, SAT_bike.exe, puede ser necesario la instalación de unos controles. Estos hacen posible la obtención de una gráfica para la exposición de valores. Concretamente, se trata del 'Microsoft Chart Control para Microsoft .NET Framework 3.5'. Este se puede descargar de forma gratuita en la página web:

<http://www.microsoft.com/es-ES/download/details.aspx?id=14422>.

1.2 INTERFAZ

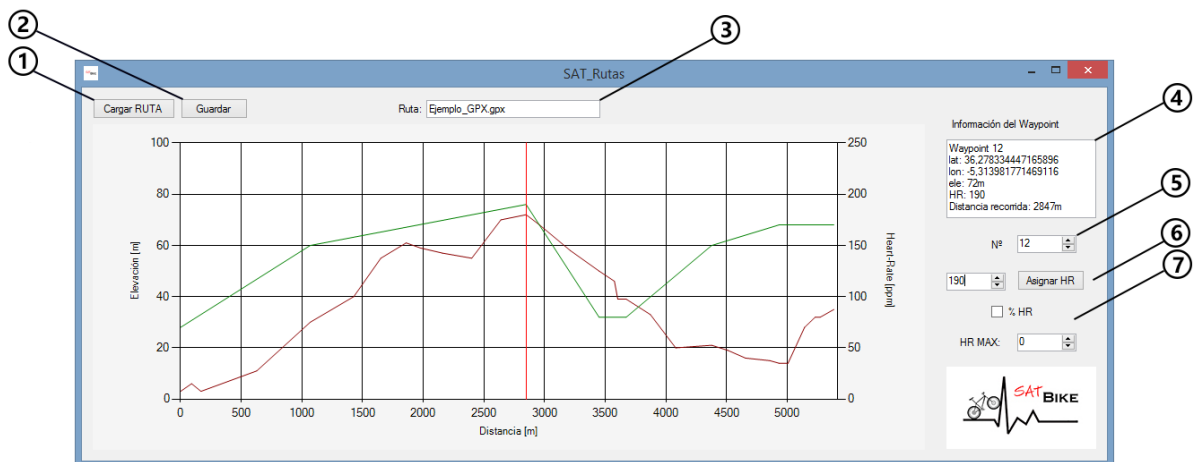


FIGURA 22. PARTES INTERFAZ SAT_RUTAS



1.3 FUNCIONAMIENTO

1. Cargar ruta: Permite buscar en el directorio la ruta en extensión gpx.
2. Guardar: Exporta en formato texto, los parámetros de la ruta. La ruta se deberá guardar bajo el nombre SATB.txt.
3. Nombre de la ruta en estudio.
4. Información de Waypoint: En esta ventana de texto se exponen los datos de cada punto que forma la ruta.
 - Latitud
 - Longitud
 - Elevación
 - Frecuencia cardíaca asignada
 - Distancia

5. Selector Waypoint: Con este selector se puede escoger el punto determinado para estudiar el recorrido.

6. Asignar HR: Con el valor introducido en el campo de texto se introduce el valor en BPM para el punto de paso presente en el campo de texto superior.

7. Fijar HR: Se crea esta opción para aquellos usuarios que prefieran meter valores como un porcentaje de un máximo fijado. Se debe marcar la pestaña '% HR' e introducir el máximo en HR MAX. Este máximo se calcula como:

$$\text{HR MAX} = 220 - \text{edad}$$

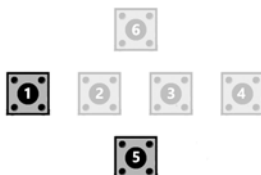
En el sector introducir el valor del porcentaje e asignar.

* Además de estos mandos, se presenta la ruta en un gráfico según su elevación. Con un cursor se posiciona en el punto según el valor que detallan 3 y 4. En rojo se dibuja la ruta, en verde los valores de frecuencia añadidos.

2. EMPLEO DE SATBIKE

Se emplearán imágenes para cada pantalla e indicarán los pulsadores activos en cada estado de color más intenso.

2.1 PANTALLA INICIAL



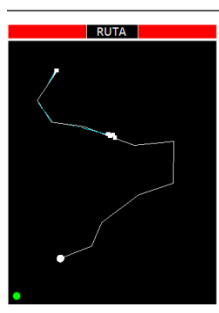
1. Transición a pantalla:

- RUTA, si existe ruta en tarjeta SD
- MON_sR, en el caso de modo libre

5. Traspasa los valores de la última ruta desde la EEPROM a un fichero en la tarjeta de memoria. Aparecerá el mensaje de 'Transfer OK', cuando se realiza correctamente.

FIGURA 23. INICIAL + PULSADORES

2.2 PANTALLA RUTA



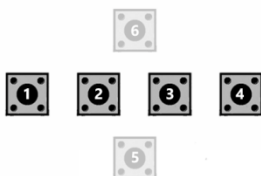
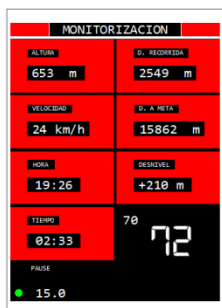
1. Transición a pantalla MON

6. Dibuja en color rojo el recorrido guardado hasta el momento.

- GPS activo y recibiendo localización
- GPS desconectado

FIGURA 24. RUTA + PULSADORES

2.3 PANTALLA MONITORIZACIÓN (MON)



1. Transición a pantalla RUTA

2. PLAY/PAUSE: Se da comienzo al cronómetro, empieza la ruta. Se empiezan a recoger valores.

3. STOP: Resetea el cronómetro

4. Transición a la pantalla HRM

- GPS activo y recibiendo localización
- GPS desconectado

FIGURA 25. MONITORIZACIÓN + PULSADORES

2.4 PANTALLA HRM

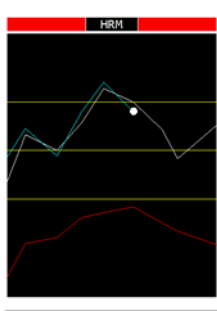
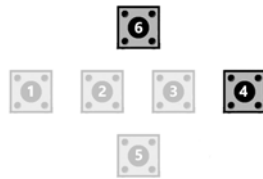


FIGURA 26. HRM + PULSADORES



4. Transición a pantalla MON

6. Dibuja en color azul claro la variación de frecuencia cardiaca guardada hasta el momento.

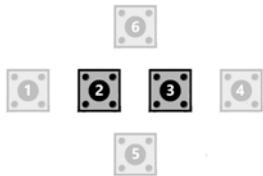
En amarillo se muestran los ejes de 50, 100 y 150 pulsaciones por minuto.

En color rojo se expone el altura para ver la correlación entre las dos gráficas.

2.5 PANTALLA MONITORIZACIÓN SIN RUTA (MON_SR)



FIGURA 27. MONITORIZACION SIN RUTA + PULSADORES



2. PLAY/PAUSE: Se da comienzo al cronómetro, empieza la ruta. Se empiezan a recoger valores.

3. STOP: Resetea el cronómetro

En esta pantalla no se muestra la distancia a meta (como en MON) sino una variable que presenta la velocidad media en

km/h.

- GPS activo y recibiendo localización
- GPS desconectado

V. CÓDIGO

1.SATBIKE.INO

```
// SATBike Julio 2014
// Iñigo Fernández de Araoz García Miñaur
// Universidad Pontifica Comillas - ICAI
// -----
/*****
    GESTIÓN PRINCIPAL Programa
    *****/

// LIBRERIAS ARDUINO
#include <ctype.h>
#include <math.h>
#include <avr/eeprom.h>
#include <EEPROMex.h>
#include <stdlib.h>
#include <SoftwareSerial.h>

// LIBRERIAS PERIFERICOS
#include <UTFT.h> // Pantalla LCD
#include <Adafruit_GPS.h> // GPS
#include <SD.h> // Tarjeta de memoria
#include "DHT.h" // Temperatura
#include "Arduino.h"

// PANTALLA
extern uint8_t BigFont[];
extern uint8_t SmallFont[];
extern uint8_t SevenSegNumFont[];
UTFT myGLCD(TFT01_22SP, 46, 44, 47, 45, 42);

// GPS
Adafruit_GPS GPS(&Serial1);
#define GPSECHO false
boolean usingInterrupt = false;
static uint32_t timer = millis();
static float lat0, lon0; // Coordenadas del punto donde comienza la ruta, una vez que
                           comienza el crono

// SENSOR PULSO
int pulsePin = 1; // Pulse Sensor purple wire connected to analog pin 0
int blinkPin = 13; // pin to blink led at each beat
volatile int BPM; // used to hold the pulse rate
volatile int Signal; // holds the incoming raw data
volatile int IBI = 600; // holds the time between beats, must be seeded!
volatile boolean Pulse = false; // true when pulse wave is high, false when it's low
volatile boolean QS = false; // becomes true when Arduino finds a beat.

// SENSOR TEMP
#define SENST A0 //Recoge el valor de la temperatura
#define DHTTYPE DHT11
DHT dht(SENST, DHTTYPE);

/*-----*/
/**
 * PROTOTIPOS
 */
/*-----*/
// PROTOTIPOS Pantallas
boolean PANT_INICIAL(int estado);
void PANT_MONITOR(float distancia);
float PANT_RUTA(int estado);
void PANT_HRM(int activo);
// PROTOTIPOS Operaciones GPS
float cDMinToDDeg (float degMin);
float HaversineM(float lat1, float lon1, float lat2, float lon2);
int Calcula_WPOINT(float lat_pos, float lon_pos, int total);
```

```
// PROTOTIPOS GPS
void useInterrupt(boolean);
// PROTOTIPOS EEPROM - SD
void SD_REALIZADO(void);
// PROTOTIPOS Sensor BPM
void InterruptSetupBMP();
// PROTOTIPOS Cronometro
void config_chrono(void);
void reset_chrono(void);
uint32_t read_microseconds(void);

/*-----*/
/** VARIABLES GLOBALES **/
/*-----*/
# define NUM 50
//Alberga los valores de la ruta, extrae de la SD
float matriz_num[NUM],matriz_lat[NUM],matriz_lon[NUM],matriz_alt[NUM],matriz_hr[NUM];
int wpoint[NUM]; //Vector que sirve para albergar la posicion de inicio de cada WAYPOINT
int wp_act; //Señala el WPOINT actual mediante la funcion
float desnivel=0, altitud; // Inicializacion de variables MON
float distancia;
int i3; // Tomará el valor del num. de waypoints

//EEPROM
float latreal,lonreal;
int address=10;
byte value;
static boolean eeeprom_ok=false;

//SD
File root;
File archivo;
static int i;
String nombre;
String seleccion;

// PULSADORES
// Pines
const int boton_1 = 22;
const int boton_2 = 23;
const int boton_3 = 24;
const int boton_4 = 25;
const int boton_5 = 26;
const int boton_6 = 27;
// Detectores de flanco independientes
static int valor_act, valor_ant, valor_ant2, valor_act2;
static int valor_act3, valor_ant3, valor_act4, valor_ant4;
static int valor_act5, valor_ant5, valor_act6, valor_ant6;

// MAQUINA de ESTADO
static int estado;
boolean ruta, ini=true; // Detector de ruta
boolean cambio;
// Controles del CRONOMETRO
static boolean play=0,pause,var_stop,punto_0=1;

/*****
***** SETUP *****/
*****/
void setup()
{
    Serial.begin(9600);

    // PANTALLA
    myGLCD.InitLCD(0);
    myGLCD.setFont(BigFont);
    estado=0;

    // GPS
    GPS.begin(9600);
    GPS.sendCommand(PMTK_SET_NMEA_OUTPUT_RMCGGA);
    GPS.sendCommand(PMTK_SET_NMEA_UPDATE_1HZ); // Actualiza cada 1Hz
    useInterrupt(true);

    // SD
    pinMode(53, OUTPUT);
```

```
//SENSOR PULSO
InterruptSetupBMP(); // Prepara el timer para 2ms

//TEMP
dht.begin();

// BOTONES
pinMode(boton_1, INPUT);
pinMode(boton_2, INPUT);
pinMode(boton_3, INPUT);
pinMode(boton_4, INPUT);
pinMode(boton_5, INPUT);
pinMode(boton_6, INPUT);

// CRONOMETRO
reset_chrono();
config_chrono();
}

/***** LOOP *****/
void loop()
{
    if(ini==1){
        ruta=PANT_INICIAL(0);
        ini=0;
    }
    // Detector flanco B1
    valor_ant=0;
    valor_act=digitalRead(boton_1);
    if((valor_ant != valor_act) && (valor_act == HIGH)){
        estado=estado+1;
    }
    valor_ant=valor_act;

    if(ruta){
        switch(estado){
            case 0:
                // NO OCURRE NADA
                break;
            case 1:
                distancia=PANT_RUTA(0);
                break;
            case 2:
                PANT_MONITOR(distancia);
                estado=0;
                break;
        }
    }else{
        PANT_MONITOR_sinR();
    }
}
```

```
/**
 * Función: PANT_INICIAL
 * Parametros: int estado
 * Devuelve: boolean arch
 * Comentarios: Inicia el sistema y discrimina modo ruta o libre
 *               Comprueba el estado de GPS
 *               Da la opción de acceder a SD_REALIZADO
 */
boolean PANT_INICIAL(int estado){
    boolean arch;
    int cant_arch, puls;
    float t;

    myGLCD.setFont(BigFont);
    myGLCD.fillScr(255,0,0);
    myGLCD.setColor(255, 255, 255);
    myGLCD.fillRect(0, 16, 240, 18);
```

```
myGLCD.fillRect(0, 280, 240, 282);
myGLCD.print("SATBike", CENTER, 0);
myGLCD.fillRect(30, 40, 210, 70);
myGLCD.print("Bienvenido", CENTER, 50);

    if (!SD.begin(53)) {
        // No se ha conectado correctamente
    }else{
        // Lectura de archivos
        root = SD.open("/");
        File entry=root.openNextFile();
        i=1;
        arch=0;
        while(entry) {
            entry = root.openNextFile();
            i=i+1;

            if(SD.exists("SATB.txt")){
                arch=1;
                myGLCD.print("RUTA", CENTER, 140);
                myGLCD.print("encontrada", CENTER, 165);
            }
        }
        if(arch!=1){
            myGLCD.print("NO RUTA", CENTER, 140);
        }

        cant_arch=i;
        i=1;
        puls=0;
        while(1){
            if (GPS.newNMEAreceived()) {
                if (!GPS.parse(GPS.lastNMEA())) ; // this also sets the newNMEAreceived()
                flag to false
            }
            if (timer > millis()) timer = millis();
            if (millis() - timer > 1000) {
                timer = millis();
                if(GPS.fix!=0){
                    myGLCD.print("SAT_OK", CENTER, 220);
                }
                myGLCD.printNumI(GPS.hour+2, 10, 295);
                myGLCD.print(":", 40, 295);
                myGLCD.printNumI(GPS.minute, 55, 295);
                myGLCD.printNumI(GPS.day, 97, 295);
                myGLCD.print("/", 127, 295);
                myGLCD.printNumI(GPS.month, 142, 295);
            }

            t = dht.readTemperature();
            myGLCD.printNumF(t, 1, 170, 295);

            // Cierre archivo;
            root.close();
            delay(50);

            // Detector flanco EEPROM - SD
            valor_ant5=0;
            valor_act5=digitalRead(boton_5);
            if((valor_ant5 != valor_act5) && (valor_act5 == HIGH)){
                SD_REALIZADO();
            }
            valor_ant5=valor_act5;

            // Poder acceder a otras pantallas
            if(digitalRead(boton_1)){
                break;
            }
        }
        // Devuelve indicador ruta
        return arch;
    }
}
```

```
/* ****
*** Función: PANT_RUTA
*** Parametros: int estado
*** Devuelve: float tdist
*** Comentarios: Dibuja la ruta:
***                 Posiciona al usuario
***                 Opción de mostrar el trayecto efectuado
**** */

// Factores de escala para el dibujo
# define factx 220
# define facty 285

float PANT_RUTA(int estado){
    char caracter[2000];
    boolean menosmax=1, menosmin=1, menos;
    int maxval, a, dist;
    int contini, i, i2, coma;
    //Permite dibujar la posicion instantanea
    float pos[1];
    float maxlat, minlat, maxlon, minlon, tdist;
    float lat_ee, lon_ee, lat_ee2, lon_ee2;
    int cont_maxlat, cont_minlat, cont_maxlon, cont_minlon;
    int cont_c_maxlat, cont_c_minlat, cont_c_maxlon, cont_c_minlon;

    // Inicializaciones
    coma=0;
    i=0; i2=0;
    contini=0;
    myGLCD.clrScr();
    myGLCD.setColor(255, 0, 0);
    myGLCD.fillRect(0, 0, 240, 16);
    myGLCD.setFont(BigFont);
    myGLCD.setColor(255, 255, 255);
    myGLCD.fillRect(0, 16, 240, 18);
    myGLCD.print("RUTA", CENTER, 0);
    archivo=SD.open("SATB.txt", FILE_READ);

    if (archivo) {
        // Si existe
        while (archivo.available()) {
            caracter[i]=archivo.read();

            if(caracter[i]==' ' && contini==0){
                cont_maxlat=i;
                contini=contini+1;
                caracter[i]=archivo.read();
            }

            if(caracter[i]==',' && coma==0){
                cont_c_maxlat=i;
                coma=1;
                caracter[i]=archivo.read();
            }

            if(caracter[i]==' ' && contini==1){
                cont_minlat=i;
                contini=contini+1;
                caracter[i]=archivo.read();
            }

            if(caracter[i]==',' && coma==1){
                cont_c_minlat=i;
                coma=2;
                caracter[i]=archivo.read();
            }

            if(caracter[i]==' ' && contini==2){
                cont_maxlon=i;
                contini=contini+1;
                caracter[i]=archivo.read();
            }

            if(caracter[i]==',' && coma==2){
                cont_c_maxlon=i;
                coma=3;
            }
        }
    }
}
```

```

        caracter[i]=archivo.read();
    }

    if(caracter[i]==' ' && contini==3){
        cont_minlon=i;
        contini=contini+1;
        caracter[i]=archivo.read();
    }

    if(caracter[i]==',' && coma==3){
        cont_c_minlon=i;
        coma=4;
        caracter[i]=archivo.read();
    }

    if(caracter[i]=='T'){
        dist=i;
    }

    if(caracter[i]=='W'){
        wpoint[i2]=i;
        i2++;
        caracter[i]=archivo.read();
    }

    ++i;
}
// Cierra el archivo
archivo.close();

maxval=i;

if(cont_c_maxlat-cont_maxlat==2){
    maxlat=(caracter[cont_maxlat]-48)*pow(10,1)+(caracter[cont_maxlat+1]-
48)+(caracter[cont_maxlat+2]-48)*pow(10,-1)+(caracter[cont_maxlat+3]-48)*pow(10,-
2)+(caracter[cont_maxlat+4]-48)*pow(10,-3)+(caracter[cont_maxlat+5]-48)*pow(10,-
4)+(caracter[cont_maxlat+6]-48)*pow(10,-5)+(caracter[cont_maxlat+7]-48)*pow(10,-
6)+(caracter[cont_maxlat+8]-48)*pow(10,-7);
} else if(cont_c_maxlat-cont_maxlat==1){
    maxlat=(caracter[cont_maxlat]-48)+(caracter[cont_maxlat+2]-48)*pow(10,-
1)+(caracter[cont_maxlat+3]-48)*pow(10,-2)+(caracter[cont_maxlat+4]-48)*pow(10,-
3)+(caracter[cont_maxlat+5]-48)*pow(10,-4)+(caracter[cont_maxlat+6]-48)*pow(10,-
5)+(caracter[cont_maxlat+7]-48)*pow(10,-6)+(caracter[cont_maxlat+8]-48)*pow(10,-7);
}

if(cont_c_minlat-cont_minlat==2){
    minlat=(caracter[cont_minlat]-48)*pow(10,1)+(caracter[cont_minlat+1]-
48)+(caracter[cont_minlat+2]-48)*pow(10,-1)+(caracter[cont_minlat+3]-48)*pow(10,-
2)+(caracter[cont_minlat+4]-48)*pow(10,-3)+(caracter[cont_minlat+5]-48)*pow(10,-
4)+(caracter[cont_minlat+6]-48)*pow(10,-5)+(caracter[cont_minlat+7]-48)*pow(10,-
6)+(caracter[cont_minlat+8]-48)*pow(10,-7);
} else if(cont_c_maxlat-cont_maxlat==1){
    minlat=(caracter[cont_minlat]-48)+(caracter[cont_minlat+2]-48)*pow(10,-
1)+(caracter[cont_minlat+3]-48)*pow(10,-2)+(caracter[cont_minlat+4]-48)*pow(10,-
3)+(caracter[cont_minlat+5]-48)*pow(10,-4)+(caracter[cont_minlat+6]-48)*pow(10,-
5)+(caracter[cont_minlat+7]-48)*pow(10,-6)+(caracter[cont_minlat+8]-48)*pow(10,-7);
}
if(caracter[cont_maxlon]=='0') menosmax=0;

if(menosmax){
    maxlon=(-1)*((caracter[cont_maxlon+1]-48)+(caracter[cont_maxlon+2]-48)*pow(10,-
1)+(caracter[cont_maxlon+3]-48)*pow(10,-2)+(caracter[cont_maxlon+4]-48)*pow(10,-
3)+(caracter[cont_maxlon+5]-48)*pow(10,-4)+(caracter[cont_maxlon+6]-48)*pow(10,-
5)+(caracter[cont_maxlon+7]-48)*pow(10,-6)+(caracter[cont_maxlon+8]-48)*pow(10,-7));
} else {
    maxlon=(caracter[cont_maxlon+1]-48)+(caracter[cont_maxlon+2]-48)*pow(10,-
1)+(caracter[cont_maxlon+3]-48)*pow(10,-2)+(caracter[cont_maxlon+4]-48)*pow(10,-
3)+(caracter[cont_maxlon+5]-48)*pow(10,-4)+(caracter[cont_maxlon+6]-48)*pow(10,-
5)+(caracter[cont_maxlon+7]-48)*pow(10,-6)+(caracter[cont_maxlon+8]-48)*pow(10,-7);
}

if(caracter[cont_minlon]=='0') menosmin=0;

if(menosmin){
    minlon=(-1)*((caracter[cont_minlon+1]-48)+(caracter[cont_minlon+2]-48)*pow(10,-
1)+(caracter[cont_minlon+3]-48)*pow(10,-2)+(caracter[cont_minlon+4]-48)*pow(10,-
3)+(caracter[cont_minlon+5]-48)*pow(10,-4)+(caracter[cont_minlon+6]-48)*pow(10,-

```

```

5)+(caracter[cont_minlon+7]-48)*pow(10,-6)+(caracter[cont_minlon+8]-48)*pow(10,-7));
    }else {
        minlon=(caracter[cont_minlon+1]-48)+(caracter[cont_minlon+2]-48)*pow(10,-
1)+(caracter[cont_minlon+3]-48)*pow(10,-2)+(caracter[cont_minlon+4]-48)*pow(10,-
3)+(caracter[cont_minlon+5]-48)*pow(10,-4)+(caracter[cont_minlon+6]-48)*pow(10,-
5)+(caracter[cont_minlon+7]-48)*pow(10,-6)+(caracter[cont_minlon+8]-48)*pow(10,-7);
    }

    tdist=(caracter[dist+7]-48)*pow(10,4)+(caracter[dist+8]-
48)*pow(10,3)+(caracter[dist+9]-48)*pow(10,2)+(caracter[dist+10]-
48)*pow(10,1)+(caracter[dist+11]-48);

    i3=i2;

    for(i=0;i<i2;++i){
        matriz_num[i]=i;
        a=wpoint[i];
        matriz_lat[i]=(caracter[a+12]-48)*pow(10,1)+(caracter[a+13]-48)+(caracter[a+15]-
48)*pow(10,-1)+(caracter[a+16]-48)*pow(10,-2)+(caracter[a+17]-48)*pow(10,-
3)+(caracter[a+18]-48)*pow(10,-4)+(caracter[a+19]-48)*pow(10,-5)+(caracter[a+20]-
48)*pow(10,-6);

        if(caracter[a+25]=='-'){
            matriz_lon[i]=(-1)*((caracter[a+26]-48)+(caracter[a+28]-48)*pow(10,-
1)+(caracter[a+29]-48)*pow(10,-2)+(caracter[a+30]-48)*pow(10,-3)+(caracter[a+31]-
48)*pow(10,-4)+(caracter[a+32]-48)*pow(10,-5)+(caracter[a+33]-48)*pow(10,-6));
        }else{
            matriz_lon[i]=(caracter[a+26]-48)+(caracter[a+28]-48)*pow(10,-
1)+(caracter[a+29]-48)*pow(10,-2)+(caracter[a+30]-48)*pow(10,-3)+(caracter[a+31]-
48)*pow(10,-4)+(caracter[a+32]-48)*pow(10,-5)+(caracter[a+33]-48)*pow(10,-6);
        }

        matriz_alt[i]=(caracter[a+38]-48)*pow(10,3)+(caracter[a+39]-
48)*pow(10,2)+(caracter[a+40]-48)*pow(10,1)+(caracter[a+41]-48);
        matriz_hr[i]=(caracter[a+44]-48)*pow(10,2)+(caracter[a+45]-
48)*pow(10,1)+(caracter[a+46]-48);
    }
    // Fin if(archivo)
}

/*
if(play){
    // Opcion de dibujar por defecto

    for(i=2;i<=address/10+1;i++){
        lat_ee=EEPROM.readFloat(10*(i-1));
        lon_ee=EEPROM.readFloat(10*(i-1)+4);
        lat_ee2=EEPROM.readFloat(10*i);
        lon_ee2=EEPROM.readFloat(10*i+4);
        myGLCD.setColor(0,255,255);
        myGLCD.drawLine(5+(lon_ee-minlon)/(maxlon-minlon)*factx,30+(maxlat-
lat_ee)/(maxlat-minlat)*facty,5+(lon_ee2-minlon)/(maxlon-minlon)*factx,30+(maxlat-
lat_ee2)/(maxlat-minlat)*facty);
        myGLCD.setColor(255,255,255);
    }

}*/

// VERIFICACION
// Serial.println(maxlat);
// Serial.println(minlat);
// Serial.println(maxlon);
// Serial.println(minlon);
// Serial.println(tdist);

    if(maxlon<0){
        myGLCD.fillRect(5+(matriz_lon[0]-minlon)/(maxlon-minlon)*factx+3,30+(maxlat-
matriz_lat[0])/(maxlat-minlat)*facty+3,5+(matriz_lon[0]-minlon)/(maxlon-minlon)*factx-
3,30+(maxlat-matriz_lat[0])/(maxlat-minlat)*facty-3);
        myGLCD.fillCircle(5+(matriz_lon[i2-1]-minlon)/(maxlon-minlon)*factx,30+(maxlat-
matriz_lat[i2-1])/(maxlat-minlat)*facty,3);
        for(i=1;i<i2;++i){
            myGLCD.drawLine(5+(matriz_lon[i-1]-minlon)/(maxlon-minlon)*factx,30+(maxlat-
matriz_lat[i-1])/(maxlat-minlat)*facty,5+(matriz_lon[i]-minlon)/(maxlon-
minlon)*factx,30+(maxlat-matriz_lat[i])/(maxlat-minlat)*facty);
        }
    }else{

```

```

Serial.println("Entra en positivo");
myGLCD.fillRect(5+(matriz_lon[0]-minlon)/(maxlon-minlon)*factx+3,30+(maxlat-
matriz_lat[0])/(maxlat-minlat)*facty+3,5+(matriz_lon[0]-minlon)/(maxlon-minlon)*factx-
3,30+(maxlat-matriz_lat[0])/(maxlat-minlat)*facty-3);
myGLCD.fillCircle(5+(matriz_lon[i2-1]-minlon)/(maxlon-minlon)*factx,30+(maxlat-
matriz_lat[i2-1])/(maxlat-minlat)*facty,3);
for(i=1;i<i2;++i){
    myGLCD.drawLine(5+(matriz_lon[i-1]-minlon)/(maxlon-minlon)*factx,30+(maxlat-
matriz_lat[i-1])/(maxlat-minlat)*facty,5+(matriz_lon[i]-minlon)/(maxlon-
minlon)*factx,30+(maxlat-matriz_lat[i])/(maxlat-minlat)*facty);
}
}

// WHILE
while(1){

    if (GPS.newNMEAreceived()) {
        if (!GPS.parse(GPS.lastNMEA())) ; // this also sets the newNMEAreceived() flag
to false
    }

    if (timer > millis()) timer = millis();
    // Cada segundo se obtiene info
    if (millis() - timer > 1000) {
        timer = millis();
        pos[0]=cDMinToDDeg(GPS.latitude);

        pos[1]=(-1)*cDMinToDDeg(GPS.longitude);

        if(GPS.fix!=0){
            myGLCD.setColor(0,255,0);
            myGLCD.fillCircle(10,310,3);
            myGLCD.setColor(255,255,255);
            // CRONO & EEPROM
            latreal=cDMinToDDeg(GPS.latitude);
            lonreal=(-1)*cDMinToDDeg(GPS.longitude);
            if(play){
                if(GPS.altitude>altitud){
                    desnivel=desnivel+(GPS.altitude-altitud);
                }else if(GPS.altitude<altitud){
                    desnivel=desnivel-(altitud-GPS.altitude);
                }
                altitud=GPS.altitude;
            }else{
                altitud=GPS.altitude;
            }
        }else{
            myGLCD.setColor(255,0,0);
            myGLCD.fillCircle(10,310,3);
            myGLCD.setColor(255,255,255);
        }
    }

    if(maxlon<0){
        Serial.println(pos[0],6);
        Serial.println(pos[1],6);
        myGLCD.fillRect(5+(pos[1]-minlon)/(maxlon-minlon)*factx+3,30+(maxlat-
pos[0])/(maxlat-minlat)*facty+3,5+(pos[1]-minlon)/(maxlon-minlon)*factx-3,30+(maxlat-
pos[0])/(maxlat-minlat)*facty-3);
        //myGLCD.fillRect(5+(-3.73-minlon)/(maxlon-minlon)*factx+3,30+(maxlat-
40.4)/(maxlat-minlat)*facty+3,5+(-3.73-minlon)/(maxlon-minlon)*factx-3,30+(maxlat-
40.4)/(maxlat-minlat)*facty-3);
        Serial.println("ENTRO A APROBAR");
    }else{
        myGLCD.fillRect(5+(pos[1]-minlon)/(maxlon-minlon)*factx+3,30+(maxlat-
pos[0])/(maxlat-minlat)*facty+3,5+(pos[1]-minlon)/(maxlon-minlon)*factx-3,30+(maxlat-
pos[0])/(maxlat-minlat)*facty-3);
    }
}

if((seconds == 30 || seconds == 0) && play){
    if(EEPROM.isReady() && eeprom_ok==true){
        EEPROM.writeInt(8,address/10);
        EEPROM.writeFloat(address,latreal);
        address=address+4;
        EEPROM.writeFloat(address,lonreal);
    }
}

```

```

        address=address+4;
        // Verificacion
        /* Serial.print("LATITUD: ");
        Serial.println(latreal);
        Serial.print("LONGITUD: ");
        Serial.println(lonreal);
        Serial.print("HR: ");
        Serial.println(BPM); */

        if(BPM>35){
            EEPROM.writeInt(address,BPM);
        }else{
            EEPROM.writeInt(address,35);
        }
        address=address+2;

        eeprom_ok=false;
    }

    if(eeprom_ok==false && ((seconds>5 && seconds<10) || (seconds>35 && seconds<40))){
        //Entra a cambiar el valor de control de la EEPROM";
        eeprom_ok=true;
    }

    valor_ant6=0;
    valor_act6=digitalRead(boton_6);
    if((valor_ant6 != valor_act6) && (valor_act6 == HIGH)){
        // DIBUJA LO QUE LLEVO
        for(i=2;i<address/10+1;i++){
            if(address==10){
            }else{
                lat_ee=EEPROM.readFloat(10*(i-1));
                lon_ee=EEPROM.readFloat(10*(i-1)+4);
                lat_ee2=EEPROM.readFloat(10*i);
                lon_ee2=EEPROM.readFloat(10*i+4);
                myGLCD.setColor(255,0,0);
                myGLCD.drawLine(5+(lon_ee-minlon)/(maxlon-minlon)*factx,30+(maxlat-lat_ee)/(maxlat-minlat)*facty,5+(lon_ee2-minlon)/(maxlon-minlon)*factx,30+(maxlat-lat_ee2)/(maxlat-minlat)*facty);
                myGLCD.setColor(255,255,255);
            }
        }
        valor_ant6=valor_act6;

        if(digitalRead(boton_1)){
            break;
        }
    }
    return tdist;
}

```

```

/*****
*** Función: PANT_MONITOR
*** Parametros: float distancia
*** Devuelve: -
*** Comentarios: Obteniendo la distancia hace las funciones:
***                 Recibir GPS
***                 Gestión del cronómetro
***                 Recepción de sensores
*****/

void PANT_MONITOR(float distancia){
    boolean dos_activo=false;
    int min_display, sec_display,min_act=0;
    int vez,i4=0, punto=0;
    float latact,lonact;
    float fecha,hora;
    float drec,drest;
    float t;
}

```

```

myGLCD.clrScr();
myGLCD.setColor(255, 0, 0);
myGLCD.fillRect(0, 0, 240, 16);
myGLCD.fillRect(0,20, 118, 80);
myGLCD.fillRect(0,84, 118, 144);
myGLCD.fillRect(0,148, 118, 212);
myGLCD.fillRect(0, 216, 118, 276);
myGLCD.fillRect(122,20, 240, 80);
myGLCD.fillRect(122,84, 240, 144);
myGLCD.fillRect(122,148, 240, 212);

myGLCD.setColor(255, 255, 255);
myGLCD.fillRect(0, 16, 240, 18);
myGLCD.setFont(BigFont);
myGLCD.print("MONITORIZACION", CENTER, 0);
myGLCD.setFont(SmallFont);
myGLCD.print("ALTITUD", 20,30);
myGLCD.print("VELOCIDAD", 20,94);
myGLCD.print("HORA", 20,158);
myGLCD.print("TIEMPO", 20,226);
myGLCD.print("D. RECORRIDA", 137,30);
myGLCD.print("D. A META", 142,94);
myGLCD.print("DESNIVEL", 142,158);

myGLCD.print(" km/h",60,114);
myGLCD.print(" m",210,114);
myGLCD.setFont(BigFont);
myGLCD.print(" m",60,50);
myGLCD.print(" m",185,50);
myGLCD.print(" m",185,178);
myGLCD.printNumI(distancia,135,114);

while(1){
    if (GPS.newNMEAreceived()) {
        // Entra en la recepcion de señal

    if (!GPS.parse(GPS.lastNMEA())) ;
        // Descifra la trama y compara con anterior
    }

    if (timer > millis()) timer = millis();
    if (millis() - timer > 5000) {
        timer = millis();

        // Temperatura
        t = dht.readTemperature();
        myGLCD.printNumF(t,1,30,300);

        if(GPS.fix!=0){
            myGLCD.setColor(0,255,0);
            myGLCD.fillCircle(10,310,3);
            myGLCD.setColor(255,255,255);

            // CRONO & EEPROM
            latreal=cDMinToDDeg(GPS.latitude);
            lonreal=(-1)*cDMinToDDeg(GPS.longitude);
            if(play){
                if(GPS.altitude>altitud){
                    desnivel=desnivel+(GPS.altitude-altitud);
                }else if(GPS.altitude<altitud){
                    desnivel=desnivel-(altitud-GPS.altitude);
                }
            }
            altitud=GPS.altitude;

            // Tras obtener valores se exponen
            myGLCD.setFont(BigFont);
            myGLCD.setColor(255, 255, 255);
            myGLCD.printNumI(GPS.altitude,15,50);
            myGLCD.printNumI(GPS.speed*1.852*3.6,15,114);
            myGLCD.printNumI(GPS.hour+2,10,178);
            myGLCD.print(":",40,178);
            myGLCD.printNumI(GPS.minute,55,178);
            myGLCD.print(":",40,246);

        }else{

```

```
// No se recibe señal
myGLCD.setColor(255,0,0);
myGLCD.fillCircle(10,310,3);
myGLCD.setColor(255,255,255);
}

// CRONO Detector flanco
valor_ant2=0;
valor_act2=digitalRead(boton_2);
if((valor_ant2 != valor_act2) && (valor_act2 == HIGH)){
    dos_activo=true;
    // Punto inicial de la ruta
    if(punto_0){
        altitud=GPS.altitude;
        lat0=cDMinToDDeg(GPS.latitude);
        lon0=(-1)*cDMinToDDeg(GPS.longitude);
        fecha=GPS.day*pow(10,4)+GPS.month*pow(10,2)+GPS.year;
        hora=(GPS.hour+2)*pow(10,2)+GPS.minute;
        EEPROM.writeFloat(0,fecha);
        EEPROM.writeFloat(4,hora);
        punto_0=false;
    }

    if(dos_activo==true && play==true){
        play=!play;
        myGLCD.setFont(SmallFont);
        myGLCD.print("PAUSE", 20,280);
        myGLCD.setFont(BigFont);
        delay(500);
        myGLCD.setColor(0, 0, 0);
        myGLCD.fillRect(0, 277, 118, 290);
        myGLCD.setColor(255, 255, 255);
        dos_activo=false;
    }
    if(dos_activo==true && play==false){
        play=!play;
        eeeprom_ok=true; // Importante para ir guardando datos
        myGLCD.setFont(SmallFont);
        myGLCD.print("PLAY", 20,280);
        myGLCD.setFont(BigFont);
        delay(500);
        myGLCD.setColor(0, 0, 0);
        myGLCD.fillRect(0, 277, 118, 290);
        myGLCD.setColor(255, 255, 255);
        dos_activo=false;
    }
}
// Fin flanco
}
valor_ant2=valor_act2;

valor_ant3=0;
valor_act3=digitalRead(boton_3);
if((valor_ant3 != valor_act3) && (valor_act3 == HIGH)){
    reset_chrono();
    play=false;
    var_stop=true;
    punto_0=true;
    myGLCD.print("00", 55, 246);
    myGLCD.print("00", 10, 246);
    myGLCD.setFont(SmallFont);
    myGLCD.print("STOP", 20, 280);
    delay(500);
    myGLCD.setColor(0, 0, 0);
    myGLCD.fillRect(0, 277, 118, 290);
    myGLCD.setColor(255, 255, 255);
}
valor_ant3=valor_act3;

// Guardar en EEPROM
if((seconds == 30 || seconds == 0) && play){
    if(EEPROM.isReady() && eeeprom_ok==true){
        EEPROM.writeInt(8,address/10);
        EEPROM.writeFloat(address,latreal);
        address=address+4;
    }
}
```

```

        EEPROM.writeFloat(address, lonreal);
        address=address+4;
        if(BPM>35){
            EEPROM.writeInt(address, BPM);
        }else{
            EEPROM.writeInt(address, 35);
        }
        address=address+2;
        eeprom_ok=false;
    }
}
if(eeprom_ok==false &&((seconds>5 && seconds<10) || (seconds>35 && seconds<40))){
    // ENTRA A CAMBIAR EL VALOR de CONTROL EEPROM");
    eeprom_ok=true;
}

// Actualización de while
min_display=(int)minutes;
sec_display=(int)seconds;
if(play){
    punto=Calcula_WPOINT((float)cDMinToDDeg(GPS.latitude),
(float)cDMinToDDeg(GPS.longitude), i3);
    myGLCD.printNumI(matriz_hr[punto], 125, 216);
    if(sec_display%10==0){
        myGLCD.setFont(BigFont);
        drec=HaversineM(lat0, lon0, cDMinToDDeg(GPS.latitude), (-
1)*cDMinToDDeg(GPS.longitude));
        drest=distancia-drec;
        myGLCD.printNumI(drec, 137, 50);
        myGLCD.printNumI(drest, 135, 114);
        myGLCD.printNumI(desnivel, 138, 178);
    }

    if(min_display!=min_act){
        min_act=min_display;
    }

    if(sec_display<10){
        myGLCD.print("0", 54, 246);
        myGLCD.printNumI(sec_display, 70, 246);
    }else{
        myGLCD.printNumI(sec_display, 55, 246);
    }

    if(min_display<10){
        myGLCD.printNumI(min_display, 25, 246);
    }else{
        myGLCD.printNumI(min_display, 10, 246);
    }
}

// Recoge valor sensor
if(QS == true){
    myGLCD.setFont(SevenSegNumFont);
    if(BPM<100){
        if(vez==1){
            myGLCD.setColor(0, 0, 0);
            myGLCD.fillRect(120, 240, 230, 300);
            myGLCD.setColor(255, 255, 255);
            vez=0;
        }

        myGLCD.printNumI(BPM, 160, 250);
    }else{
        myGLCD.printNumI(BPM, 130, 250);
        vez=1;
    }
    myGLCD.setFont(BigFont);
    // reset del conteo de maximos
    QS = false;
}

valor_ant4=0;
valor_act4=digitalRead(boton_4);
if((valor_ant4 != valor_act4) && (valor_act4 == HIGH)){
    PANT_HRM(0);
}

```

```

        valor_ant4=valor_act4;

        if(digitalRead(boton_1)){
            break;
        }
    }
    return;
}

```

```

/*****
*** Función: PANT_HRM
*** Parametros: int activo
*** Devuelve: -
*** Comentarios: Dibuja el perfil de corazon & altura:
*** Opción de mostrar la variación del perfil efectuado
*****/

void PANT_HRM(int activo){
    char caracter[2000];
    float dist[50];
    dist[0]=0;
    int hecho1,hecho2,hechoa,hechob;
    int punto;
    float x1=0,y1,x2=0,y2;
    float act,bct;
    int maxval, a;
    int i,i2,coma,contini;
    boolean menosmax=1,menosmin=1,menos;

    coma=0;
    i=0; i2=0;
    contini=0;
    myGLCD.clrScr();
    myGLCD.setColor(255, 0, 0);
    myGLCD.fillRect(0, 0, 240, 16);
    myGLCD.setFont(BigFont);
    myGLCD.setColor(255, 255, 255);
    myGLCD.fillRect(0, 16, 240, 18);
    myGLCD.print("HRM", CENTER,0);
    archivo=SD.open("SATB.txt",FILE_READ);

    if (archivo) {
        //myGLCD.print("Tira", CENTER,130);
        // read from the file until there's nothing else in it:
        while (archivo.available()) {

            caracter[i]=archivo.read();

            if(caracter[i]=='W'){
                wpoint[i2]=i;
                i2++;
                caracter[i]=archivo.read();
            }
            //Serial.println(caracter[i]);
            //if(caracter[i]==' '){
            // Serial.println(caracter[i]);
            //}
            ++i;
        }
        // close the file:
        archivo.close();
        maxval=i;

        i3=i2;
        for(i=0;i<=i2-2;++i){
            if(i==i2-2){
                ++i;
                a=wpoint[i];
                act=(caracter[a+44]-48)*pow(10,2)+(caracter[a+45]-
48)*pow(10,1)+(caracter[a+46]-48);
                y1=205-(act-35)/185*205+20;
                myGLCD.drawLine(x2,y2,240,y1);
                --i;
            }else{

```

```

a=wpoint[i];
act=(caracter[a+44]-48)*pow(10,2)+(caracter[a+45]-48)*pow(10,1)+(caracter[a+46]-
48);
//x1=240*i/i2;
if(i==0) x1=0;
else{
x1+=240*HaversineM(matriz_lat[i],matriz_lon[i],matriz_lat[i+1],matriz_lon[i+1])/dis
tancia;
}
y1=205-(act-35)/185*205+20;
++i;
a=wpoint[i];

bct=(caracter[a+44]-48)*pow(10,2)+(caracter[a+45]-48)*pow(10,1)+(caracter[a+46]-
48);
x2=x1+240*HaversineM(matriz_lat[i],matriz_lon[i],matriz_lat[i+1],matriz_lon[i+1])/d
istancia;
y2=205-(bct-35)/185*205+20;
myGLCD.drawLine(x1,y1,x2,y2);
--i;
}
dist[i+1]=dist[i]+HaversineM(matriz_lat[i],matriz_lon[i],matriz_lat[i+1],matriz_lon[
i+1]);
myGLCD.setColor(255,0,0);
myGLCD.drawLine(240*dist[i]/distancia,300-(matriz_alt[i]-
600),240*dist[i+1]/distancia,300-(matriz_alt[i+1]-600));
myGLCD.setColor(255,255,255);
}
// EJES
myGLCD.setColor(255,255,0);
myGLCD.drawLine(0,208,240,208); //50 BPM
myGLCD.drawLine(0,152,240,152); //100 BPM
myGLCD.drawLine(0,97,240,97); //150 BPM
myGLCD.setColor(255,255,255);
//myGLCD.drawLine(x1,y1,240,y1);

}
while(1){
if (GPS.newNMEAreceived()) {
    Serial.println("Entra en el nuevo seg");
    // a tricky thing here is if we print the NMEA sentence, or data
    // we end up not listening and catching other sentences!
    // so be very wary if using OUTPUT_ALLDATA and trying to print out data
    //Serial.println(GPS.lastNMEA()); // this also sets the newNMEAreceived() flag to false

    if (!GPS.parse(GPS.lastNMEA())) ; // this also sets the newNMEAreceived() flag to false
    // we can fail to parse a sentence in which case we should just wait for another
}

if (timer > millis()) timer = millis();
if (millis() - timer > 1000) {
    timer = millis();
    if(GPS.fix!=0){
        latreal=cDMinToDDeg(GPS.latitude);
        lonreal=(-1)*cDMinToDDeg(GPS.longitude);
        if(GPS.altitude>altitud){
            desnivel=desnivel+(GPS.altitude-altitud);
        }else if(GPS.altitude<altitud){
            desnivel=desnivel-(altitud-GPS.altitude);
        }
        altitud=GPS.altitude;
        punto=Calcula_WPOINT((float)cDMinToDDeg(GPS.latitude),
(float)cDMinToDDeg(GPS.longitude),i3);
        Serial.print("//");
        Serial.print(punto);
        Serial.print("//");
        Serial.println();
    }
    // myGLCD.drawLine(0,300-alt_hr[0][1]*300/240,100,300-alt_hr[1][1]*300/240);
    if(BPM>35){
        myGLCD.fillCircle(240*punto/i3,300-(float)BPM/240*300,3);
    }else{
        myGLCD.fillCircle(240*punto/i3,300-(float)35/240*300,3);
    }
}

}

if((seconds == 30 || seconds == 0) && play){
    if(EEPROM.isReady() && eeprom_ok==true){

```

```

        EEPROM.writeInt(8,address/10);
        EEPROM.writeFloat(address,latreal);
        address=address+4;
        EEPROM.writeFloat(address,lonreal);
        address=address+4;
        if(BPM>35){
            EEPROM.writeInt(address,BPM);
        }else{
            EEPROM.writeInt(address,35);
        }
        address=address+2;
        eeprom_ok=false;
        // Dibujo lo que llevo recorrido
    }
}
if(eeprom_ok==false && ((seconds>5 && seconds<10) || (seconds>35 &&
seconds<40))){
    Serial.println("ENTRA A CAMBIAR EL VALOR de CONTROL EEPROM");
    eeprom_ok=true;
}

valor_ant6=0;
valor_act6=digitalRead(boton_6);
if((valor_ant6 != valor_act6) && (valor_act6 == HIGH)){
    for(i=1;i<=80/10;i++){
        if(address==10){
        }else{
            if(i==address/10){
            }else{
                myGLCD.setColor(0, 255, 255);
                Serial.println("EEEEEEEntrAAAAA");
                hecho1=240*dist[i-1]/distancia;
                hechoa=205-((float)EEPROM.readInt(18+10*(i-1))-35)/185*205+20;
                hecho2=240*dist[i]/distancia;
                hechob=205-((float)EEPROM.readInt(18+10*i)-35)/185*205+20;
                Serial.println(hechoa);
                Serial.println(hechob);
                myGLCD.drawLine(hecho1,hechoa,hecho2,hechob);
                Serial.println(hecho1);
                Serial.println(hecho2);
                myGLCD.setColor(255, 255, 255);
            }
        }
    }
}
valor_ant6=valor_act6;

if(digitalRead(boton_4)){
    myGLCD.clrScr();
    myGLCD.setColor(255, 0, 0);
    myGLCD.fillRect(0, 0, 240, 16);
    myGLCD.fillRect(0,20, 118, 80);
    myGLCD.fillRect(0,84, 118, 144);
    myGLCD.fillRect(0,148, 118, 212);
    myGLCD.fillRect(0, 216, 118, 276);
    myGLCD.fillRect(122,20, 240, 80);
    myGLCD.fillRect(122,84, 240, 144);
    myGLCD.fillRect(122,148, 240, 212);

    myGLCD.setColor(255, 255, 255);
    myGLCD.fillRect(0, 16, 240, 18);
    myGLCD.setFont(BigFont);
    myGLCD.print("MONITORIZACION",CENTER,0);
    myGLCD.setFont(SmallFont);
    myGLCD.print("ALTITUD", 20,30);
    myGLCD.print("VELOCIDAD", 20,94);
    myGLCD.print("HORA", 20,158);
    myGLCD.print("TIEMPO", 20,226);
    myGLCD.print("D. RECORRIDA", 137,30);
    myGLCD.print("D. A META", 142,94);
    myGLCD.print("DESNIVEL", 142,158);
    myGLCD.print(" km/h",30,114);
    myGLCD.print(" m",185,114);
    myGLCD.setFont(BigFont);
    myGLCD.print(" m",185,50);
}

```

```
myGLCD.print(" m",185,178);  
myGLCD.print(" m",60,50);  
myGLCD.printNumI(distancia,135,114);  
  
    break;  
}  
}  
return;  
}
```

```
/*  
*** Función: SD_REALIZADO  
*** Parametros: -  
*** Devuelve: -  
*** Comentarios: Traspasa la última información de la EEPROM  
creando un archivo en la tarjeta SD y ordenando los valores  
*****/  
  
void SD_REALIZADO(void){  
    int cant_wp, i_sd;  
    float fecha;  
    File arch;  
    String nombre,extension;  
    char buffer[10];  
    char cadena[20];  
  
    fecha=EEPROM.readFloat(0);  
    cant_wp=EEPROM.readInt(8);  
    // Crea el nombre del archivo obtiene fecha  
    nombre=dtostrf(fecha,6,0,buffer);  
    extension=String(".txt");  
    nombre.concat(extension);  
    nombre.toCharArray(cadena,20);  
  
    // Abre la SD y crea el archivo  
    arch = SD.open(cadena,FILE_WRITE);  
    if(arch){  
        for(i_sd=1;i_sd<=cant_wp;i_sd++){  
            arch.print("WAYpoint ");  
            arch.println(i_sd-1);  
            arch.println(EEPROM.readFloat(10*i_sd),6);  
            arch.println(EEPROM.readFloat(10*i_sd+4),6);  
            arch.println(EEPROM.readInt(10*i_sd+8));  
        }  
        myGLCD.print("Transfer OK",CENTER,92);  
    }  
    // Cierra la tarjeta SD para luego poder abrir  
    arch.close();  
    return;  
}
```

```
// Interrupción por milisegundo consulta señal GPS  
SIGNAL(TIMER0_COMPA_vect) {  
    char c = GPS.read();  
  
    if (GPSECHO)  
        if (c) UDR0 = c;  
    // Escribir en UDR0 es mucho más rápido que Serial  
}  
  
void useInterrupt(boolean v) {  
    if (v) {  
        OCR0A = 0xFF;  
        TIMSK0 |= _BV(OCIE0A);  
        usingInterrupt = true;  
    } else {  
        TIMSK0 &= ~_BV(OCIE0A);  
    }  
}
```

```
usingInterrupt = false;
}
}

// VARIABLES CRONOMETRO
uint32_t microseconds_count = 0;
uint16_t milliseconds_display = 0, microseconds_display = 0, milliseconds_rest = 0;
uint16_t microseconds = 0, milliseconds = 0;
uint8_t seconds = 0, minutes = 0, hours = 0;

// Timer-1 OC1A Control
ISR(TIMER1_COMPA_vect)
{
    cli();
    if(play){
        milliseconds += 100;
        if (milliseconds == 1000){
            milliseconds = 0;
            seconds++;
        }

        if (seconds == 60){
            seconds = 0;
            minutes++;
        }
        if (minutes == 60){
            minutes = 0;
            hours++;
        }
    }
    sei();
}

void config_chrono(void)
{
    TCCR1A = 1<<WGM10 | 1<<WGM11;
    TCCR1B = 1<<CS10 | 1<<CS11 | 1<<WGM12 | 1<<WGM13; // F_CPU=16MHz/64 (preescala)
    OCR1A = 24999; // Cuenta a 24999
    TIMSK1 = 1<<OCIE1A; // Enable interrupt cuando coincidan
    sei(); // Enable interrupts
}

// Reset chronometer
void reset_chrono(void)
{
    unsigned char sreg;
    sreg = SREG; // Guarda el flag
    cli(); // Desajusta interrupciones
    TCNT1 = 0; // Resetea el timer
    SREG = sreg;
    milliseconds = 0;
    seconds = 0;
    minutes = 0;
    hours = 0;
}

// Read microseconds counter
uint32_t read_microseconds(void)
{
    uint32_t microseconds;
    unsigned char sreg;
    sreg = SREG; // Guarda el flag
    cli(); // Disable interrupts
    microseconds = TCNT1*4;
    SREG = sreg;
    return microseconds;
}

float cDMinToDDeg (float degMin) {
    float min = 0.0;
    float decDeg = 0.0;

    //obtiene los minutos
    min = fmod((float)degMin, 100.0);
```

```
//conforma las coordenadas
degMin = (int) ( degMin / 100 );
decDeg = degMin + ( min / 60 );

return decDeg;
}

float HaversineM(float lat1,float lon1,float lat2,float lon2){
    //Calcula la distancia entre dos puntos a partir de su LAT y LON
    float dlong, dlat;
    float a, c, d;
    float d2r, eQRadius;
    d2r = PI / 180;
    eQRadius = 6378.137;
    dlong = (lon2 - lon1) * d2r;
    dlat = (lat2 - lat1) * d2r;
    a = pow(sin(dlat / 2), 2) + cos(lat1 * d2r) * cos(lat2 * d2r) * pow(sin(dlong / 2),
2);
    c = 2 * atan2(sqrt(a), sqrt(1 - a));
    d = eQRadius*c*1000;
    return d;
}

int Calcula_WPOINT(float lat_pos, float lon_pos,int total){
    int wpoint=0;
    float min_dist,dist;

    min_dist=HaversineM(lat_pos,lon_pos,matriz_lat[0],matriz_lon[0]);
    for (i=0;i<total;i++){
        dist=HaversineM(lat_pos,lon_pos,matriz_lat[i],matriz_lon[i]);
        if(dist<min_dist){
            min_dist=dist;
            wpoint=i;
        }
    }
    return wpoint;
}
```

```
// Interrupcion sensor
volatile int rate[10]; // array to hold last ten IBI values
volatile unsigned long sampleCounter = 0; // used to determine pulse timing
volatile unsigned long lastBeatTime = 0; // used to find IBI
volatile int P = 512; // used to find peak in pulse wave, seeded
volatile int T = 512; // used to find trough in pulse wave, seeded
volatile int thresh = 512; // used to find instant moment of heart beat,
seeded
volatile int amp = 100; // used to hold amplitude of pulse waveform,
seeded
volatile boolean firstBeat = true; // used to seed rate array so we startup with
reasonable BPM
volatile boolean secondBeat = false; // used to seed rate array so we startup with
reasonable BPM

void InterruptSetupBMP(){
    // Initializes Timer2 to throw an interrupt every 2mS.
    TCCR2A = 0x02; // DISABLE PWM ON DIGITAL PINS 3 AND 11, AND GO INTO CTC MODE
    TCCR2B = 0x06; // DON'T FORCE COMPARE, 256 PRESCALER
    OCR2A = 0x7C; // SET THE TOP OF THE COUNT TO 124 FOR 500Hz SAMPLE RATE
    TIMSK2 = 0x02; // ENABLE INTERRUPT ON MATCH BETWEEN TIMER2 AND OCR2A
    sei(); // MAKE SURE GLOBAL INTERRUPTS ARE ENABLED
}

// THIS IS THE TIMER 2 INTERRUPT SERVICE ROUTINE.
// Timer 2 makes sure that we take a reading every 2 miliseconds
ISR(TIMER2_COMPA_vect){
    cli(); // triggered when Timer2 counts to 124
    // disable interrupts while we do this
    //Serial.println("lee");
    Signal = analogRead(pulsePin); // read the Pulse Sensor
    sampleCounter += 2; // keep track of the time in mS with this
variable
```

```

int N = sampleCounter - lastBeatTime;           // monitor the time since the last beat to
avoid noise

// find the peak and trough of the pulse wave
if(Signal < thresh && N > (IBI/5)*3){           // avoid dichrotic noise by waiting 3/5 of
last IBI
    if (Signal < T){                           // T is the trough
        T = Signal;                           // keep track of lowest point in pulse wave
    }
}

if(Signal > thresh && Signal > P){               // thresh condition helps avoid noise
    P = Signal;                               // P is the peak
}                                              // keep track of highest point in pulse wave

// NOW IT'S TIME TO LOOK FOR THE HEART BEAT
// signal surges up in value every time there is a pulse
if (N > 250){                                  // avoid high frequency noise
    if ( (Signal > thresh) && (Pulse == false) && (N > (IBI/5)*3) ){
        Pulse = true;                         // set the Pulse flag when we think there
is a pulse
        digitalWrite(blinkPin,HIGH);          // turn on pin 13 LED
        IBI = sampleCounter - lastBeatTime;   // measure time between beats in mS
        lastBeatTime = sampleCounter;         // keep track of time for next pulse

        if(secondBeat){                       // if this is the second beat, if secondBeat ==
TRUE
            secondBeat = false;               // clear secondBeat flag
            for(int i=0; i<=9; i++){           // seed the running total to get a realistic
BPM at startup
                rate[i] = IBI;
            }
        }

        if(firstBeat){                        // if it's the first time we found a beat, if
firstBeat == TRUE
            firstBeat = false;                // clear firstBeat flag
            secondBeat = true;                // set the second beat flag
            sei();                             // enable interrupts again
            return;                            // IBI value is unreliable so discard it
        }

        // keep a running total of the last 10 IBI values
        word runningTotal = 0;                // clear the runningTotal variable

        for(int i=0; i<=8; i++){               // shift data in the rate array
            rate[i] = rate[i+1];               // and drop the oldest IBI value
            runningTotal += rate[i];           // add up the 9 oldest IBI values
        }

        rate[9] = IBI;                        // add the latest IBI to the rate array
        runningTotal += rate[9];               // add the latest IBI to runningTotal
        runningTotal /= 10;                   // average the last 10 IBI values
        BPM = 60000/runningTotal;             // how many beats can fit into a minute?
that's BPM!
        QS = true;                            // set Quantified Self flag
        // QS FLAG IS NOT CLEARED INSIDE THIS ISR
    }
}

if (Signal < thresh && Pulse == true){          // when the values are going down, the beat is
over
    digitalWrite(blinkPin,LOW);               // turn off pin 13 LED
    Pulse = false;                           // reset the Pulse flag so we can do it again
    amp = P - T;                             // get amplitude of the pulse wave
    thresh = amp/2 + T;                       // set thresh at 50% of the amplitude
    P = thresh;                               // reset these for next time
    T = thresh;
}

if (N > 2500){                                 // if 2.5 seconds go by without a beat
    thresh = 512;                             // set thresh default
    P = 512;                                  // set P default
    T = 512;                                  // set T default
    lastBeatTime = sampleCounter;              // bring the lastBeatTime up to date
    firstBeat = true;                          // set these to avoid noise
}

```

```

    secondBeat = false;                                // when we get the heartbeat back
}
sei();                                                  // enable interrupts when youre done!
} // end isr

```

2. SAT_RUTAS.VB

El diseño de la interfaz tiene la apariencia que se muestra en la Figura 28. El nombre de los elementos que componen este diseño se añade a continuación:

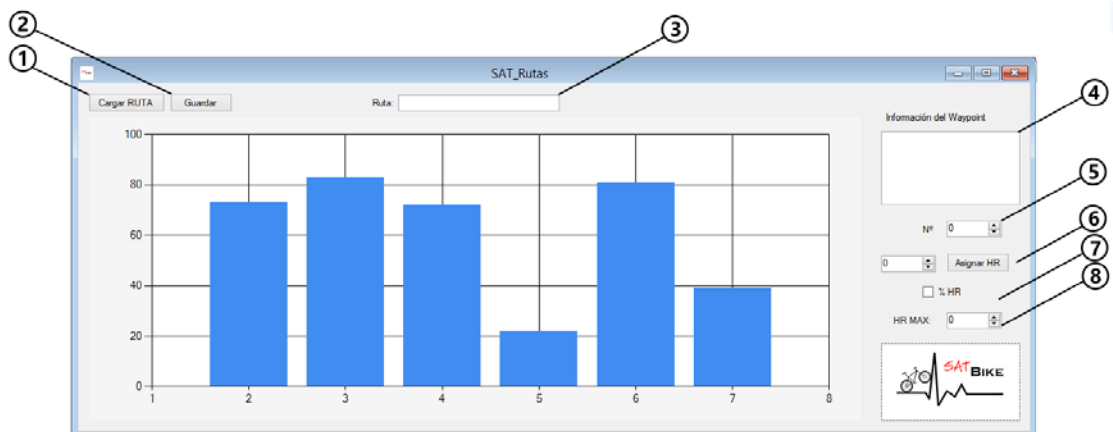


FIGURA 28. DISEÑO INTERFAZ SAT_RUTAS

1. BtnOpen (Button)
2. Btn_Save (Button)
3. Nom_RUTA (TextBox)
4. TextBox1 (TextBox)
5. Num_CHK (NumericUpDown)
6. Asig_HRM (NumericUpDown)
7. Btn_Asignar (Button)
8. PorcentajeTICK (CheckBox)
9. MAXporc (NumericUpDown)

```

' SATBike Julio 2014
' Iñigo Fernández de Araoz García Miñaur
' Universidad Pontificia Comillas - ICAI
' -----

Imports System.IO
Imports System.Globalization
Public Class SATBike
    'Variables
    'Apertura de archivo
    Dim FileReader As StreamReader
    Dim results As DialogResult

```

```
'Control lectura de .gpx
Dim linea, nombre As String
Dim chq_nombre As String
Dim chequeo1, chequeo3, chequeosalida As Boolean
Dim num_ini, num_ini_ele, num_fin_ele As Integer
Dim num_chq1, num_chq1b, num_chq2, num_chq2b, num_chq3, num_chq3b As Integer
'Permite la lectura de signos negativos y puntos decimales
Dim style As Globalization.NumberStyles
'Obtencion de datos en matrices
'MAXIMO 1000 Waypoints
Dim numero(1000, 4), grafica(1000, 1) As Double
Dim matriz(1000, 4)
Dim cant As Decimal
'Calculo de la distancia entre puntos
Dim distancia, delta_dist As Double
Dim intervalox, intervaloy As Integer
'Variables escalar
Dim min_y, max_y As Double
Dim max_ax_y, min_ax_y As Double

' Subrutina: BtnOpen_Click
' Comentarios: Permite abrir el directorio de archivos y escoger una ruta GPX
'               Expone el nombre de la ruta en la ventana superior
'               Extrae los valores de la ruta y los guarda en variables globales
'               Dibuja el perfil de altitud de la ruta
'               Muestra en el cuadro de texto los detalles de cada waypoint
' -----
Private Sub BtnOpen_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles BtnOpen.Click

    style = NumberStyles.AllowDecimalPoint
    results = OpenFileDialog1.ShowDialog

    If results = DialogResult.OK Then
        FileReader = New StreamReader(OpenFileDialog1.FileName)
        Nom_RUTA.Text = Limpiar(OpenFileDialog1.FileName)
        'Inicializaciones para un nuevo archivo
        ReDim matriz(1000, 4)
        cant = 0
        chequeosalida = 0
        While (chequeosalida = 0)
            linea = FileReader.ReadLine
            'Linea de latitud y longitud
            chequeo1 = linea.Contains("<" & "t" & "r" & "k" & "p" & "t" & " " & "l")
            'Linea de elevacion
            chequeo3 = linea.Contains("<" & "e")
            'Linea fin archivo
            chequeosalida = linea.Contains("<" & "/" & "g" & "p")
            If chequeo1 Then
                matriz(cant, 0) = cant
                numero(cant, 0) = cant
                num_ini = InStr(linea, "<")
                num_chq1 = InStr(linea, "=") + 1
                num_chq1b = InStrRev(linea, "l") - 2
                num_chq2 = InStrRev(linea, "=") + 1
                num_chq2b = InStrRev(linea, ">") - 2

                'VALORES LATITUD
                matriz(cant, 1) = linea.Remove(0, num_chq1).Remove(num_chq1b - num_chq1 -
1, num_chq2b - num_chq1b + 3).Replace(".", ",")
                numero(cant, 1) = Double.Parse(matriz(cant, 1), NumberStyles.Float)
                'VALORES LONGITUD
                matriz(cant, 2) = linea.Remove(0, num_chq2).Remove(num_chq2b - num_chq2,
2).Replace(".", ",")
                numero(cant, 2) = Double.Parse(matriz(cant, 2), NumberStyles.Float)
            End If
            'VALORES ELEVACIÓN
            If chequeo3 Then
                num_ini_ele = InStr(linea, "<")
                num_chq3 = InStr(linea, ">")
                num_chq3b = InStrRev(linea, "<") - 1
                num_fin_ele = InStr(linea, ">")
                matriz(cant, 3) = linea.Remove(0, num_chq3).Remove(num_chq3b - num_chq3,
6).Replace(".", ",")
                If matriz(cant, 3) <> "" Then
                    numero(cant, 3) = Double.Parse(matriz(cant, 3), style)
                Else

```

```

        numero(cant, 3) = 0
    End If
    cant = cant + 1
End If
End While

FileReader.Close()

'INICIALIZACIONES GRAFICA
Chart1.Series.Clear()
Chart1.Series.Add("Recorrido")
Chart1.Series("Recorrido").Points.Clear()
Chart1.Series("Recorrido").ChartType =
DataVisualization.Charting.SeriesChartType.Line
Chart1.Series("Recorrido").Color = Color.DarkRed
Chart1.Series.Add("Freq_HR")
Chart1.Series("Freq_HR").Points.Clear()
Chart1.Series("Freq_HR").ChartType =
DataVisualization.Charting.SeriesChartType.Line
Chart1.Series("Freq_HR").Color = Color.Green
'Calculo distancia entre puntos de RUTA
distancia = 0
For i = 0 To cant - 1
    'Llamada a la FUNCION HAVERSINE
    delta_dist = 1000D * HaversineInKM(numero(i, 1), numero(i, 2), numero(i + 1,
1), numero(i + 1, 2))
    distancia = distancia + delta_dist
    'Redondeo de la distancia a la unidad
    grafica(i + 1, 0) = Math.Round(distancia, 0)
    grafica(i, 1) = numero(i, 3)
    Chart1.Series("Recorrido").Points.AddXY(grafica(i, 0), grafica(i, 1))
    'Comienza con el eje secundario
    Chart1.Series("Freq_HR").Points.AddXY(grafica(i, 0), 0)
Next

'EJE X
Chart1.ChartAreas(0).AxisX.Minimum = 0
'Ajusta el intervalo de x en 10 redondeando
intervalox = grafica(cant - 2, 0) / 10
intervalox = Math.Round(intervalox * 0.01) * 100
Chart1.ChartAreas(0).AxisX.Interval = intervalox
Chart1.ChartAreas(0).AxisX.Title = "Distancia [m]"
'EJE Y
Chart1.Series("Freq_HR").YAxisType =
DataVisualization.Charting.AxisType.Secondary
Chart1.ChartAreas(0).AxisY.Title = "Elevación [m]"
Chart1.ChartAreas(0).AxisY2.Title = "Heart-Rate [ppm]"
Chart1.ChartAreas(0).AxisY2.MajorGrid.Enabled = False
Chart1.ChartAreas(0).AxisY2.MinorGrid.Enabled = False
'OPCIONAL - Llamada a una función escaladora de ejes
ESC_ejes(1)

'Comenzar con el texto en el primer Checkpoint
Dibujar(matriz, 0)

End If
End Sub
' Subrutina: Num_CHK_ValueChanged
' Comentarios: Permite modificar el waypoint seleccionado
'               Cambia la información de la ventana de texto
'               Actualiza la posición del cursor
' -----
Private Sub Num_CHK_ValueChanged(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Num_CHK.ValueChanged
    Num_CHK.Maximum = cant - 1
    Num_CHK.Minimum = 0
    'Modifica los datos de waypoint
    Dibujar(matriz, Num_CHK.Value)
    'Coloca el cursor en el checkpoint fijado
    Chart1.ChartAreas(0).CursorX.SetCursorPosition(grafica(Num_CHK.Value, 0))
End Sub
' Subrutina: Asig_HRM_ValueChanged
' Comentarios: Escoge los limites superior e inferior en las modalidades de porcentaje o
absoluto
'               Permite escoger entre un selector de número la frecuencia
' -----

```

```

Private Sub Asig_HRM_ValueChanged(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Asig_HRM.ValueChanged
    If PorcentajeTICK.Checked Then
        Asig_HRM.Maximum = 100
        Asig_HRM.Minimum = 0
    Else
        Asig_HRM.Maximum = 250
        Asig_HRM.Minimum = 0
    End If
End Sub
' Subrutina: PorcentajeTICK_CheckedChanged
' Comentarios: Determina el modo de introducción de valores de frecuencia porcentaje o
absoluto
' -----
Private Sub PorcentajeTICK_CheckedChanged(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles PorcentajeTICK.CheckedChanged
    Asig_HRM.Maximum = 250
    If PorcentajeTICK.Checked Then
        Asig_HRM.Maximum = 100
    End If
End Sub
' Subrutina: Btn_Asignar_Click
' Comentarios: Introduce el valor de pulsaciones planeado en la matriz 'numero'
'               Añade en el gráfico el valor asignado en color verde
'               Incorpora al texto el valor asignado
' -----
Private Sub Btn_Asignar_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles Btn_Asignar.Click
    If PorcentajeTICK.Checked Then
        numero(Num_CHK.Value, 4) = Asig_HRM.Value / 100 * MAXporc.Value
    Else
        numero(Num_CHK.Value, 4) = Asig_HRM.Value
    End If
    Chart1.Series("Freq_HR").Points.Clear()
    Dibujar(matriz, Num_CHK.Value)

    If numero(0, 4) <> 0 Then
    Else
        Chart1.Series("Freq_HR").Points.AddXY(0, 0)
    End If
    For i = 0 To cant - 1
        If numero(i, 4) <> 0 Then
            Chart1.Series("Freq_HR").Points.AddXY(grafica(i, 0), numero(i, 4))
        End If
    Next
End Sub
' Subrutina: Btn_Save_Click
' Comentarios: Genera el archivo de salida imprimiendo los valores
'               Abre el directorio con el nombre de defecto 'SATB.txt'
'               Permite guardar el archivo en la carpeta escogida
' -----
Private Sub Btn_Save_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles Btn_Save.Click
    'GENERA el archivo de texto salida
    Dim maxlon, minlon, diez As Double
    Dim FileWriter As StreamWriter
    Dim results As DialogResult
    results = SaveFileDialog1.ShowDialog
    If results = DialogResult.OK Then
        FileWriter = New StreamWriter(SaveFileDialog1.FileName, False)
        FileWriter.WriteLine("maxlat: " & Maximo(numero, 1))
        FileWriter.WriteLine("minlat: " & Minimo(numero, 1))
        maxlon = Maximo(numero, 2)
        minlon = Minimo(numero, 2)
        diez = 10
        If maxlon > 0 Then
            FileWriter.WriteLine("maxlon: 0" & maxlon)
            FileWriter.WriteLine("minlon: 0" & minlon)
        Else
            FileWriter.WriteLine("maxlon: " & maxlon)
            FileWriter.WriteLine("minlon: " & minlon)
        End If
        If grafica(cant - 1, 0) < 10 Then
            FileWriter.WriteLine("Tdist: " & "0000" & grafica(cant - 1, 0))
        ElseIf grafica(cant - 1, 0) < 100 Then

```

```

        FileWriter.WriteLine("Tdist: " & "000" & grafica(cant - 1, 0))
    ElseIf grafica(cant - 1, 0) < 1000 Then
        FileWriter.WriteLine("Tdist: " & "00" & grafica(cant - 1, 0))
    ElseIf grafica(cant - 1, 0) < 10000 Then
        FileWriter.WriteLine("Tdist: " & "0" & grafica(cant - 1, 0))
    ElseIf grafica(cant - 1, 0) < 100000 Then
        FileWriter.WriteLine("Tdist: " & grafica(cant - 1, 0))
    End If

    For i = 0 To cant - 1
        If numero(i, 0) < 10 Then
            FileWriter.WriteLine("WAYpoint" & "00" & numero(i, 0))
        ElseIf numero(i, 0) < 100 Then
            FileWriter.WriteLine("WAYpoint" & "0" & numero(i, 0))
        ElseIf numero(i, 0) < 1000 Then
            FileWriter.WriteLine("WAYpoint" & numero(i, 0))
        End If

        If Math.Round(numero(i, 1), 8) - Math.Round(numero(i, 1), 7) = 0 Then
            FileWriter.WriteLine(Math.Round(numero(i, 1), 8) & "0")
        Else
            FileWriter.WriteLine(Math.Round(numero(i, 1), 8))
        End If
        If Math.Round(numero(i, 2), 8) > 0 Then
            If Math.Round(numero(i, 2), 8) - Math.Round(numero(i, 2), 7) = 0 Then
                FileWriter.WriteLine("0" & Math.Round(numero(i, 2), 8) & "0")
            Else
                FileWriter.WriteLine("0" & Math.Round(numero(i, 2), 8))
            End If
        Else
            If Math.Round(numero(i, 2), 8) - Math.Round(numero(i, 2), 7) = 0 Then
                FileWriter.WriteLine(Math.Round(numero(i, 2), 8) & "0")
            Else
                FileWriter.WriteLine(Math.Round(numero(i, 2), 8))
            End If
        End If
        If numero(i, 3) < 10 Then
            FileWriter.WriteLine("000" & numero(i, 3))
        ElseIf numero(i, 3) < 100 Then
            FileWriter.WriteLine("00" & numero(i, 3))
        ElseIf numero(i, 3) < 1000 Then
            FileWriter.WriteLine("0" & numero(i, 3))
        Else
            FileWriter.WriteLine(numero(i, 3))
        End If
        If numero(i, 4) < 10 Then
            FileWriter.WriteLine("00" & numero(i, 4))
        ElseIf numero(i, 4) < 100 Then
            FileWriter.WriteLine("0" & numero(i, 4))
        ElseIf numero(i, 4) < 1000 Then
            FileWriter.WriteLine(numero(i, 4))
        End If
    Next
    FileWriter.Close()
End If
End Sub

' Funcion: Maximo
' Parametros: double[][] valores, int column
' Devuelve: double max_valor
' Comentarios: Calcula el máximo de la columna de una matriz
' -----
Function Maximo(ByVal valores, ByVal column)
    'Calcula el máximo de una columna
    Dim max_valor As Double
    max_valor = valores(0, column)
    For i = 1 To cant - 1
        If max_valor < valores(i, column) Then
            max_valor = valores(i, column)
        End If
    Next
    Return max_valor
End Function

' Funcion: Minimo
' Parametros: double[][] valores, int column
' Devuelve: double min_valor
' Comentarios: Calcula el mínimo de la columna de una matriz
' -----

```

```

Function Minimo(ByVal valores, ByVal column)
    'Calcula el minimo de una columna
    Dim min_valor As Double
    min_valor = valores(0, column)
    For i = 1 To cant - 1
        If min_valor > valores(i, column) Then
            min_valor = valores(i, column)
        End If
    Next
    Return min_valor
End Function

' Funcion: HaversineInKM
' Parametros: double lat1, double long1 double lat2, double long2
' Devuelve: double d
' Comentarios: Calcula la distancia en linea recta entre dos puntos
               a través de sus coordenadas
'-----
Private Function HaversineInKM(ByVal lat1, ByVal long1, ByVal lat2, ByVal long2)
    'Calcula la distancia entre dos puntos a partir de su LAT y LON
    Dim dlong, dlat As Double
    Dim a, c, d As Double
    Dim d2r, eQRadius As Double
    d2r = (Math.PI / 180D)
    eQRadius = 6378.137D
    dlong = (long2 - long1) * d2r
    dlat = (lat2 - lat1) * d2r
    a = Math.Pow(Math.Sin(dlat / 2D), 2D) + Math.Cos(lat1 * d2r) * Math.Cos(lat2 * d2r) *
Math.Pow(Math.Sin(dlong / 2D), 2D)
    c = 2D * Math.Atan2(Math.Sqrt(a), Math.Sqrt(1D - a))
    d = eQRadius * c
    Return d
End Function

' Funcion: Dibujar
' Parametros: double[][] matriz, int fila
' Devuelve: 0
' Comentarios: Expone los valores de un waypoint a traves de la matriz que los contiene
'-----
Private Function Dibujar(ByVal matriz, ByVal fila)
    'Expone los resultados en el texto
    TextBox1.Text = ""
    TextBox1.Text = "Waypoint " & fila.ToString + vbNewLine
    TextBox1.Text += "lat: " & matriz(fila, 1).ToString + vbNewLine
    TextBox1.Text += "lon: " & matriz(fila, 2).ToString + vbNewLine
    TextBox1.Text += "ele: " & numero(fila, 3).ToString & "m" + vbNewLine
    TextBox1.Text += "HR: " & numero(fila, 4).ToString + vbNewLine
    TextBox1.Text += "Distancia recorrida: " & grafica(Num_CHK.Value, 0).ToString & "m"
    Return 0
End Function

' Funcion: ESC_ejes
' Parametros: boolean opcion
' Devuelve: 0
' Comentarios: Ajusta los ejes de la gráfica en función de las características de la ruta
'-----
Function ESC_ejes(ByVal opcion)
    'Ajusta el tamaño de los ejes (ZOOM)
    If opcion = 1 Then
        'MAX
        max_y = Maximo(grafica, 1)
        max_ax_y = Math.Round((max_y + max_y * 10 / 100) * 0.01) * 100
        If max_ax_y = 0 Then
            Chart1.ChartAreas(0).AxisY.Maximum = max_y + (10 - max_y \ 10)
        Else
            Chart1.ChartAreas(0).AxisY.Maximum = max_ax_y
        End If
        Chart1.ChartAreas(0).AxisY2.Maximum = 250
        'MIN
        min_y = Minimo(grafica, 1)
        min_ax_y = Math.Round((min_y - min_y * 10 / 100) * 0.01) * 100
        Chart1.ChartAreas(0).AxisY.Minimum = min_ax_y
        Chart1.ChartAreas(0).AxisY2.Minimum = 0
    End If
    Return 0
End Function

' Funcion: Limpiar
' Parametros: string direccion
' Devuelve: string nombre

```



```
' Comentarios: Elimina la dirección del archivo, quedándose solo con el nombre de la ruta
' -----
Function Limpiar(ByVal direccion)
    'Obtains only the name of the file
    chq_nombre = InStrRev(direccion, "\")
    nombre = direccion.Remove(0, chq_nombre)
    Return nombre
End Function

'Fin del programa
End Class
```

VI. HOJAS DE CARACTERÍSTICAS Y LIBRERÍAS

1. HOJAS DE CARACTERÍSTICAS

- Arduino MEGA
<http://arduino.cc/en/Main/arduinoBoardMega2560>
- Pantalla TFT01 - 2.2SP 240 x 320 + lector SD
http://www.electfreaks.com/wiki/index.php?title=2.2S%22_TFT_LCD:_TFT01-2.2S
- GPS MTK3339 Adafruit
<http://www.adafruit.com/products/746>
- Antena GPS con base magnética
<http://php2.twinner.com.tw/files/onshine/ANT555-2006-NEW.pdf>
- DHT11 - Sensor de Temperatura
- <http://www.oddwires.com/content/DHT11.pdf>
- Sensor ritmo cardíaco - Pulse sensor Amped
<http://pulsesensor.myshopify.com/>

2. LIBRERÍAS

- uTFT
<http://henningkarlsen.com/electronics/library.php?id=52>
- EEPROMex
<http://playground.arduino.cc/Code/EEPROMex>
- Adafruit_GPS
<http://playground.arduino.cc/Code/EEPROMex>
- DHT11
<http://playground.arduino.cc/Main/DHTLib>